Bauhaus-Universität Weimar
Informatik in der Architektur | InfAR

Arbeitspapiere
Working Papers

Nr. 9, September 2011

Grundriss-Generierung mit K-Dimensionalen Baumstrukturen

Katja Knecht

ISSN 2191-2416

Bauhaus-Universität Weimar, Professur Informatik in der Architektur, Belvederer Allee 1, 99421 Weimar
Fon: +49/3643/584201, caad@architektur.uni-weimar.de, http://infar.architektur.uni-weimar.de

Katja Knecht
Generierung von Grundriss-Layouts mithilfe von Evolutionären Algorithmen und K-dimensionalen Baumstrukturen

Weimar 2011
Arbeitspapiere (Working Papers) Informatik in der Architektur, Nr. 9
Herausgegeben von Prof. Dr. Dirk Donath und Dr. Reinhard König
ISSN 2191-2416

Bauhaus-Universität Weimar, Professur Informatik in der Architektur
Belvederer Allee 1, 99425 Weimar
http://infar.architektur.uni-weimar.de

Redaktionelle Anmerkung:

Der Text ist im Rahmen des von der DFG geförderten Forschungsprojekts „KREMLAS: Entwicklung einer kreativen evolutionären Entwurfsmethode für Layoutprobleme in Architektur und Städtebau“ (DO 551/19-1) entstanden. http://infar.architektur.uni-weimar.de/service/drupal-cms/kremlas

Generierung von Grundriss-Layouts mithilfe von Evolutionären Algorithmen und K-dimensionalen Baumstrukturen

Katja Knecht
Katja.knecht@uni-weimar.de
Professur Informatik in der Architektur
Fakultät Architektur, Bauhaus-Universität Weimar, Belvederer Allee 1, 99421 Weimar, Germany

Abstract

K-dimensionale Bäume, im Englischen verkürzt auch K-d Trees genannt, sind binäre Such- und Partitionierungsbäume, die eine Menge von n Punkten in einem multidimensionalen Raum repräsentieren. Ihren Einsatz finden K-d Tree Datenstrukturen vor allem bei der Suche nach den nächsten Nachbarn, der Nearest Neighbor Query, und in weiteren Suchalgorithmen für beispielsweise Datenbankapplikationen. Im Rahmen des Forschungsprojekts Kremlas wurde die Raumpartitionierung durch K-d Trees als eine Teillösung zur Generierung von Layouts bei der Entwicklung einer kreativen evolutionären Entwurfsmethode für Layoutprobleme in Architektur und Städtebau entwickelt. Der Entwurf und die Entwicklung von Layouts, d.h. die Anordnung von Räumen, Baukörpern und Gebäudekomplexen im architektonischen und städtischen Kontext stellt eine zentrale Aufgabe in Architektur und Stadtplanung dar. Sie erfordert von Architekten und Planern funktionale sowie kreative Problemlösungen. Das Forschungsprojekt beschäftigt sich folglich nicht nur mit der Optimierung von Grundrissen sondern bindet auch gestalterische Aspekte mit ein. In der entwickelten Teillösung dient der K-d Tree Algorithmus zunächst zur Unterteilung einer vorgegebenen Fläche, wobei die Schnittlinien möglichen Raumgrenzen entsprechen. Durch die Kombination des K-d Tree Algorithmus mit genetischen Algorithmen und evolutionären Strategien werden Layouts hinsichtlich der Kriterien Raumgröße und Nachbarschaften optimiert. Durch die Interaktion des Nutzers können die Lösungen dynamisch angepasst und zur Laufzeit nach gestalterischen Kriterien verändert werden. Das Ergebnis ist ein generativer Mechanismus, der bei der kreativen algorithmischen Lösung von Layoutaufgaben in Architektur und Städtebau eine vielversprechende Variante zu bereits bekannten Algorithmen darstellt.

Keywords: Grundrissgenerierung, Multikriterielle Optimierung, Evolutionäre Algorithmen, Evolutionäre Strategie, Genetische Algorithmen, K-d Trees, Computational Design

1. Einleitung

K-dimensionale Bäume, englisch verkürzt K-d Trees genannt, sind binäre Datenstrukturen aus dem Bereich der Computergeometrie, die in zahlreichen Suchalgorithmen eingesetzt werden. Aufgrund ihrer Effizienz und ihrer Genauigkeit finden sie in vielfältigen Gebieten wie beispielsweise in Datenbankapplikationen und Raytracing-Verfahren Anwendung. Ihre Effizienz, d.h., die Geschwindigkeit mit der sie durchsucht werden können, begründet sich auf dem eingesetzten Algorithmus, mit dem sie Objekte im k-dimensionalen Raum organisieren und als Datenstruktur speichern.

Aus der algorithmischen Raumpartitionierung entsteht jedoch auch eine geometrische, parzellenhafte Struktur, deren gestalterisches Potenzial bisher wenig beachtet wurde. Im Rahmen des Forschungsprojekts zur Entwicklung einer kreativen evolutionären Entwurfsmethode für Layoutprobleme in Architektur und Städtebau (Kremlas) beschäftigten wir uns mit der Frage, wie das generative und gestalterische Potential von K-d Tree Algorithmen zur Grundrissgenerierung eingesetzt werden kann.

Der Einsatz von Algorithmen und generativen Mechanismen im Allgemeinen wird bereits seit einiger Zeit im Bereich der Architektur untersucht und insbesondere auch zur Generierung von Grundrissen eingesetzt. Erste Ansätze der computergestützten Layoutgenerierung gehen bis in die 60iger Jahre des 20. Jahrhunderts zurück (Frazer, 1974). Da der Entwurf und die Entwicklung von Layouts, d.h. die Anordnung von Räumen, Baukörpern und Gebäudekomplexen im architektonischen und städtischen Kontext, eine zentrale Aufgabe in Architektur und Stadtplanung darstellt und von Architekten und Planern funktionale sowie kreative Problemlösungen erfordert, werden diese Aufgaben häufig mit evolutionären Methoden bearbeitet (Elezkurtaj & Franck, 2002).

Das vorliegende Arbeitspapier beschreibt die Entwicklung eines generativen Mechanismus zur Grundrissgenerierung, welcher einen Raumpartitionierungsalgorithmus durch K-dimensionale Bäume als gestalterisches Mittel einsetzt und diesen mit genetischen Algorithmen und evolutionären Strategien verknüpft. Das Ergebnis stellt eine vielversprechende Variante zu bereits bekannten Algorithmen bei der kreativen algorithmischen Lösung von Layoutaufgaben in Architektur und Städtebau dar.

Dieses Arbeitspapier umfasst im folgenden Kapitel zunächst eine kurze Einführung zu k-dimensionalen Bäumen und einen kurzen Überblick über den aktuellen Stand der Forschung sowie verwandte Arbeiten im Bereich der generativen Architektur. Kapitel 3 beschreibt den Aufbau des generativen Mechanismus und des umgesetzten multikriteriellen

Optimierungssystems und stellt ferner die Interaktion des Nutzers mit dem System dar. Das Potential des eingesetzten K-d Tree Algorithmus als gestalterisches und generatives Mittel und dessen mögliche Weiterentwicklungen werden abschließend in Kapitel 4 diskutiert.

2. Stand der Forschung

2.1. Computergestützte Layoutgenerierung

Der Entwurf von Layouts stellt eine zentrale Aufgabe in Architektur und Stadtplanung dar, die sich über mehrere Maßstabsebenen zieht. Als Layoutproblem bezeichnet man im architektonischen bzw. städtischen Kontext die Anordnung von Parzellen, Gebäuden oder Räumen. Hierbei werden vom Planer funktionale und kreative Lösungen erwartet, bei deren Findung er seit einigen Jahrzehnten verstärkt durch Softwarewerkzeuge unterstützt wird. Bisherige Forschungen betrafen dabei nicht nur die Entwicklung von Zeichen-, Modellier- und Darstellungswerkzeugen sondern auch von generativen Systemen zur Unterstützung der kreativen Problemlösungsprozesse.

Die ersten Ansätze zur computergestützten Layoutgenerierung entstanden bereits in den 60iger und 70iger Jahren des 20. Jahrhunderts (Frew, 1980). Die zahlreichen Arbeiten, die seitdem in diesem Bereich entstanden sind, setzten sich mit unterschiedlichen Algorithmen, Strukturen und Modellen auseinander, um mit deren Hilfe urbane und architektonische Layouts zu generieren und den Entstehungsprozess zu automatisieren. Zu den untersuchten und eingesetzten Methoden zählen unter anderem Zelluläre Automaten, die auf städtischer und regionaler Planungsebene, beispielsweise in (Koenig & Bauriedel, 2004)(Koenig & Bauriedel, 2004), sowie auf Ebene des architektonischen Raums in (Coates et al, 1996) eingebunden wurden (Abb. 1), Lindenmeyer Systeme (Coates et al, 1999), Constraint- (Li et al, 2000) und Physically-Based Systems (Arvin & House, 2002) sowie Shape Grammars, die beispielsweise zur generativen Beschreibung von Stilen bzw. zur Generierung von Grundrissen auf Basis existierender architektonischer Vorbilder (Duarte, 2001) verwendet wurden (Abb. 2).

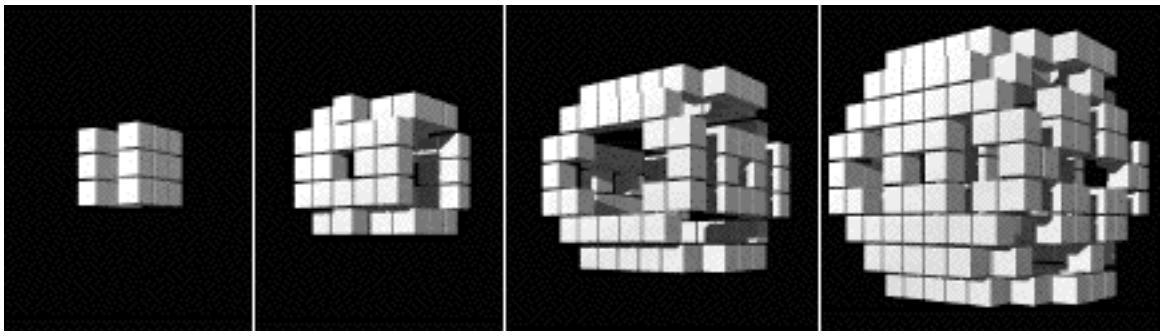


Abb. 1: Cellular Automata (Coates et al, 1996)

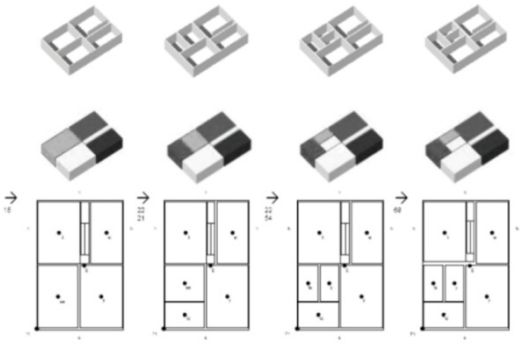


Abb. 2: Jose Duarte, Customizing Mass Housing (Duarte, 2001)

In den letzten Jahren entstanden zudem Arbeiten zu Voronoi-Diagrammen und Slicing Tree Structures, die ähnlich wie K-d Tree Strukturen durch eine Unterteilung des Raums in Unterräume entstehen. In Voronoi-Diagrammen entstehen in Abhängigkeit von der Verteilung von Objekten im Raum, den Raumzentren, durch Unterteilung sogenannte Zellen. Die Unterteilung und damit das Diagramm werden aus den Zellgrenzen gebildet, d.h. jenen Punkten, die gleichweit von mehreren Zentren entfernt liegen (De Berg et al, 1997). Es entstehen Schaumartige Strukturen und Topologien, die durch die Raumzentren und ihre Lage, Anzahl und Verteilung bestimmt werden. Voronoi-Diagramme wurden beispielsweise zur Generierung von städtebaulichen Entwürfen (Anders & König, 2011) bzw. von architektonischen Raumstrukturen (Coates et al, 2005; Harding & Derix, 2010) eingesetzt (Abb. 3).

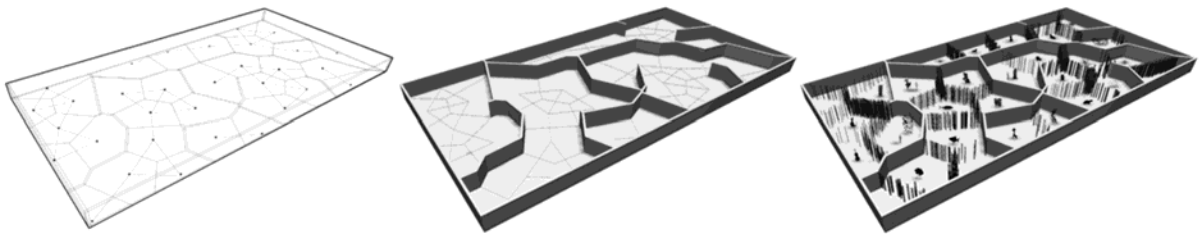


Abb. 3: Voronoi-Diagramm (Harding & Derix, 2010)

Slicing Tree Structures bauen ebenfalls auf der Methode der Raumpartitionierung durch Schnittlinien auf. Die Schnittfolge wird als Baum gespeichert, die Schnittlinien sind allerdings nicht von der Verteilung einer Punktmenge abhängig. Ihr Einsatz wurde beispielsweise zur Lösung von Facility Layout Problemen in Kombination mit Genetischen Algorithmen untersucht (Kado, 1995).

Die derzeitig vielversprechendsten Ansätze in der automatischen Layoutgenerierung bestehen aus einer Kombination von generativen Mechanismen mit evolutionären Algorithmen.

Erste Untersuchungen und Versuche zu evolutionären Algorithmen im Architekturbereich finden sich in (Frazer, 1995) und (Coates & Hazarika, 1999). Ein Entwurfsmodell zum space layout planning mit evolutionären Methoden wurde beispielsweise in (Jo & Gero, 1998) entwickelt und getestet.

Neben Genetischen Algorithmen (Kado, 1995) und Genetischer Programmierung (Coates & Hazarika, 1999; Coates et al, 1999) werden evolutionäre Algorithmen auch kombiniert wie beispielsweise bei Elezkurtaj und Franck (2002). Das entwickelte System verwendet einen Packing Algorithmus, der optimierte Layoutvorschläge in Echtzeit berechnet und ausgibt. Gleichzeitig erhält der Nutzer die Möglichkeit mit dem Layout zu interagieren und es nach seinen Vorstellungen anzupassen (Abb. 4).

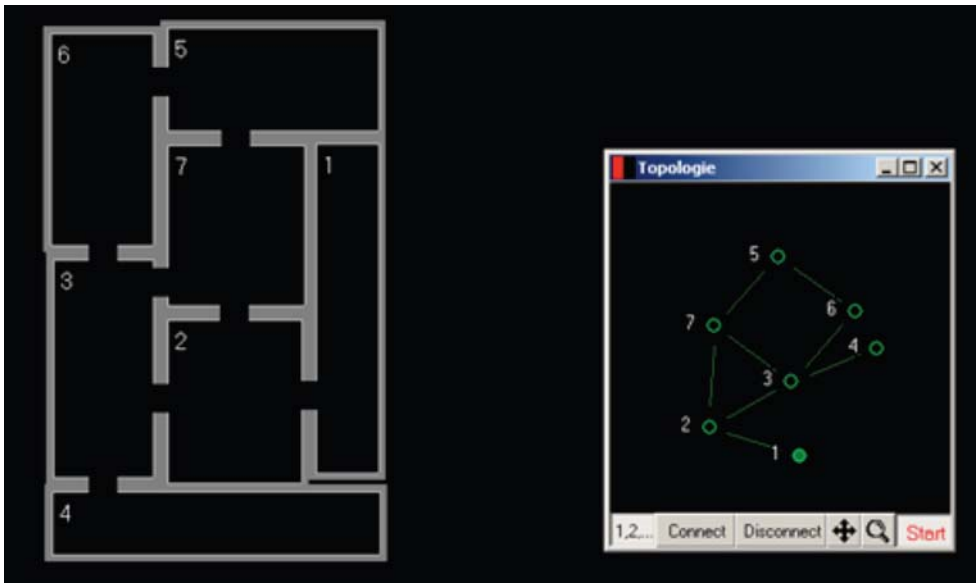


Abb. 4: Tomor Elezkurtaj, Interaktives Layout System (Elezkurtaj & Franck, 2002)

Beim Einsatz von evolutionären Methoden muss man im Allgemeinen zwischen der Optimierung vorgegebener Grundrisse und der Generierung neuer Grundrisslayouts unterscheiden (Coates et al, 1999). Die zweitgenannte Anwendung stellt hierbei die ungleich komplexere dar, da der kreative Entwurfsprozess die Lösung sowohl operationaler wie auch nicht operationaler Kriterien umfasst. Kriterien sind operational, wenn sie ein Problem eindeutig beschreiben und in ihrer Beschreibung schon den Lösungsweg aufzeigen. Nicht operationale Kriterien, wie beispielsweise ästhetische Kriterien, beschreiben hingegen vage Problemstellungen, für die es keine objektiv richtige Lösung gibt (Kirsch, 1988).

2.2. Kremlas

Im Rahmen des Forschungsprojekts Kremlas betrachten wir architektonische und städtische Layoutprobleme auf verschiedenen Maßstabsebenen, von der städtebaulichen Nachbarschaft über die Gebäudestruktur bis hin zur Grundrissorganisation. Anders als in vielen Studien, die vorwiegend auf den Einsatz von computerbasierten Hilfsmitteln zur Optimierung von Grundrissen setzen, liegt der Fokus auf der Entwicklung einer Methode, welche die kreative Komponente und die Berücksichtigung nicht operationaler Kriterien einschließt (Schneider et al, 2010).

Der Entwurfsprozess ist ein zyklischer Prozess, der aus der Betrachtung der Problemstellung, der Definition von Kriterien zur Lösung des Problems, der Generierung möglicher Lösungen, sowie deren Evaluierung und Optimierung besteht. Er zeichnet sich durch eine kontinuierliche Anpassung des Lösungsraums durch den Entwerfer im Laufe der Bearbeitung aus. Ein Entwurfssystem muss die flexible Bearbeitung verschiedener Fragestellungen ermöglichen. Top-Down und Bottom-up Strategien, d.h. Planungsaspekte mit geringer sowie stark eingeschränkter Problemdefinition, sollen beispielsweise nahtlos miteinander verknüpft werden können. In (Schneider et al, 2010) wird folgerichtig die Entwicklung eines adaptiven Entwurfssystems für Layoutprobleme vorgeschlagen, welches es dem Nutzer ermöglicht den Entwurfsraum flexibel zu definieren, anzupassen und zu durchsuchen.

Entwerfen findet darüber hinaus häufig auf verschiedenen Maßstabsebenen gleichzeitig statt. Ein System zur Lösung von Layoutproblemen sollte deshalb auf verschiedenen Maßstabsebenen adaptiv einsetzbar sein. Um den nahtlosen Übergang zwischen Maßstabsebenen im Entwurfsprozess mit möglichst wenigen Einschränkungen zu ermöglichen, wurde für die Datenstruktur des Systems eine möglichst universelle Definition von Layout gewählt. Layouts werden in diesem Sinne als die Verschachtelung von Elementen, d.h. die Anordnung von Elementen in Elementen, in Elementen usw. definiert (Abb. 5)(Schneider, Fischer, & König, 2010) (Schneider et al, 2010).

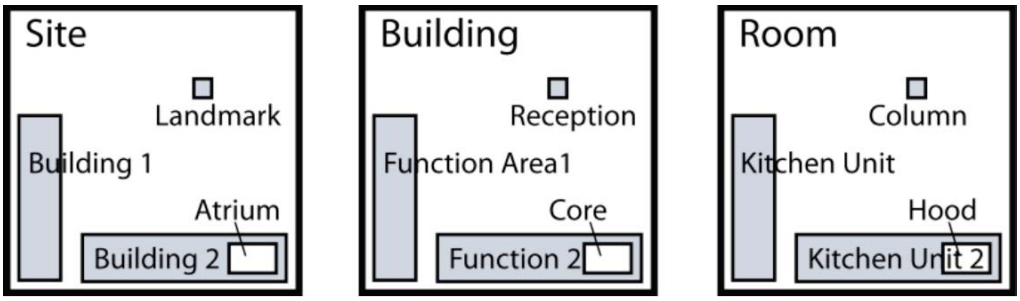


Abb. 5: Dasselbe Layout auf verschiedenen Ebenen mit unterschiedlicher Definition der Elemente (Schneider, et al., 2010)

Es ergibt sich eine hierarchische Struktur, auf deren Ebenen sich sowohl homogene als auch heterogene Elemente vermischen können. Grundsätzlich muss es möglich sein, Beziehungen zwischen Elementen herzustellen und zwar auch zwischen Elementen in verschiedenen Hierarchieebenen. Dies betrifft beispielsweise die Ausbildung notwendiger oder gewünschter Nachbarschaftsbeziehungen zwischen Elementen in horizontaler und vertikaler Abhängigkeit.

Der Entwurfsprozess ist, wie bereits erwähnt, zyklisch. Dies bedeutet, dass manche Probleme auch erst während des Prozesses erkannt werden. Daher kann das eingesetzte Entwurfssystem nicht rein linear und lösungsorientiert arbeiten, sondern es muss dem Entwerfer die Möglichkeiten bieten, die Problemstellung während des Prozesses anzupassen und neu zu definieren. Wir verfolgen daher einen explorativen Ansatz, der auf einem rekursiven, zirkulären Trial-and-Error-System aufbaut (Schneider et al, 2010).

Das System setzt generative und bewertende Mechanismen ein, die auf evolutionären Algorithmen basieren. Eine wichtige Rolle spielt bei evolutionären Algorithmen die Evaluationsfunktion, welche die generierten Lösungen nach bestimmten Kriterien beurteilt und damit die Ausrichtung der Suche im Lösungsraum bestimmt. Diese Kriterien unterscheiden sich nach Entwerfer und Entwurfsaufgabe. Daher schlagen wir die Verwendung von Kriterien vor, die so neutral und multifunktional wie möglich, sowie auf allen Maßstabsebenen gültig sind und den Elementen flexibel zugewiesen sowie durch Parameter kontrolliert werden können (Schneider et al, 2010). Hierzu gehören Kriterien wie Topologie, Orientierung, Proportion und Größe. Es können auch Kriterien zur Bewertung des Layouts eingesetzt werden, die für eine bestimmte Maßstabsebene spezifisch sind wie Dichte, Gebäudelinie, Verschattung und Anbindung.

2.3. K-d Trees

K-d Trees nach Bentley (1975) sind binäre Suchbäume, die eine Menge von n Punkten in einem multidimensionalen Raum organisieren (Bentley, 1990) und damit Datenstrukturen darstellen, die der Raumpartitionierung dienen (Goodrich & Tamassia, 2002).

Eingesetzt werden K-d Tree Datenstrukturen ursprünglich bei der Suche nach den nächsten Nachbarn, der *Nearest Neighbor Query* (Moore, 1991), und in weiteren Suchalgorithmen vor allem für klassische Datenbankapplikationen (Bentley, 1990). K-d Tree Datenstrukturen werden in Verbindung mit den genannten Suchalgorithmen aufgrund ihrer hohen Durchsuchungsgeschwindigkeit, ihrer Genauigkeit und Effizienz in vielfältigen Anwendungsgebieten eingesetzt. Zu diesen zählt beispielsweise die Gesichtserkennung im Rahmen des interakti-

ven Trackings von Gesichtszügen in Videoframes, für die Geschwindigkeit und Genauigkeit der Suchergebnisse von Relevanz ist (Buchanan & Fitzgibbon, 2006). Darüber hinaus hat sich der Einsatz von K-d Tree Datenstrukturen in Raytracing Verfahren bewährt (Fussell & Subramanian, 1988; Wald & Havran, 2006). In bisherigen Anwendungen werden K-d Trees folglich vor allem verwendet, um räumliche Daten zu organisieren und zu speichern sowie die erstellten Datenstrukturen schnell und effizient zu durchsuchen und zu durchqueren.

Eine K-d Tree Struktur wird folgendermaßen aufgebaut (siehe hierzu Tab. 1 und Abb. 6 und 7): Ausgehend von einer bekannten, finiten Punktemenge P wird der k-dimensionale Raum durch zu den Koordinatenachsen senkrechten Schnittebenen, sogenannten *partition planes*, unterteilt. Im zweidimensionalen Raum entspricht die Unterteilung einer Linie, in drei- und mehrdimensionalen Räumen handelt es sich um eine an einer Koordinatenachse ausgerichtete Hyperebene (Goodrich & Tamassia, 2002). Zur Bestimmung der Lage der Unterteilung wird aus P zunächst der Median- oder Mittelwert der Punktkoordinaten in der Schnittdimension, der *split dimension*, gebildet. Es entsteht der erste Knoten, englisch *node*, $N1$ mit dem berechneten Mittel- oder Medianwert als Schnittlinienwert, dem *split value,* der die Punktemenge P in zwei Untermengen $P1$ und $P2$, die sogenannten *sub trees*, unterteilt. Alle Punkte mit Koordinaten, die in der split dimension kleiner als das split value sind, bilden den linken Ast und sind Bestandteil von sub tree $P1$, alle Punkte mit Koordinaten die größer sind, bilden den rechten Ast und sind Bestandteil von $P2$. $N1$ gilt dabei als *root node* für die Unterknoten $N2$ und $N3$ der sub trees $P1$ und $P2$, $N2$ und $N3$ sind *child nodes* von $N1$. Diese Teilräume werden nun nach dem gleichen Prinzip weiter unterteilt, wie es für die Ausgangspunktmenge beschrieben wurde. Die Unterteilung wird so lange fortgeführt bis keine weitere Unterteilung mehr möglich oder ein bestimmter Grenzwert bzw. die vorgegebene Unterteilungstiefe erreicht ist (Abb. 6 und 7).

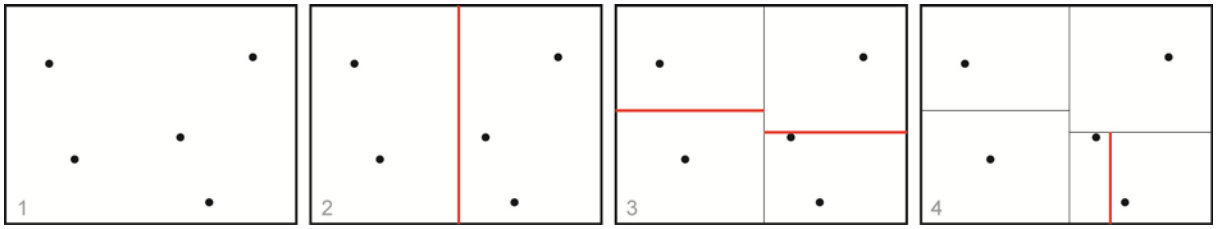


Abb. 6: Raumunterteilung durch eine K-d Tree Struktur mithilfe der Mittelwertberechnung

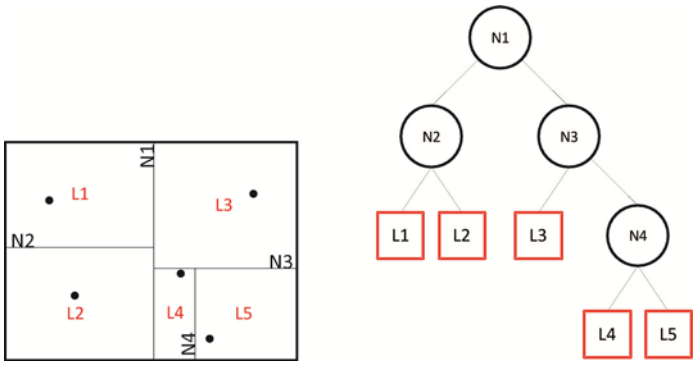


Abb. 7: Geometrische und grafische Darstellung der Unterteilung

Algorithmus:	Aufbau eines K-d Trees
Input:	Punktemenge $\boldsymbol{P}$
Output:	**kd** vom Typ **KDTree**
Logik:	1. If $\boldsymbol{P}$ ist leer return leeres **KDTree** 2. Bestimme die Schnittebene und generiere den Knoten N mit den folgenden Werten: a. **SplitDim** = die Schnittdimension b. **SplitVal** = der nach der festgelegten Schnittregel berechnete Schnittlinienwert in SplitDim 3. Fülle die rechten und linken Untermengen **Pleft** und **Pright**: a. **Pleft** = alle Punkte ϵ $\boldsymbol{P}$, mit Vektor v[SplitDim] ≤ SplitVal b. **Pright** = alle Punkte ϵ $\boldsymbol{P}$, mit Vektor v[SplitDim] > SplitVal 4. **kdleft** = linker Ast; Konstruiere das K-d Tree rekursiv mit **Pleft** 5. **kdright** = rechter Ast; Konstruiere des K-d Tree rekursiv mit **Pright** 6. return **kd**

Tab. 1: Schematischer Algorithmus des Aufbaus eines K-d Trees (Moore, 1991).

Jedem Knoten wird eine Region zugeordnet, welche wiederum Unterknoten und Unterregionen enthalten kann. Im Inneren des Suchbaums bezeichnet man die Knoten als interne Knoten, *internal nodes*. Sie unterteilen den Raum durch eine Schnittebene in einer der k Dimensionen (Bentley, 1990). Darüber hinaus werden in den inneren Knoten die Information zur Schnittebene sowie die Verweise zu den rechten und linken Unterknoten abgelegt. Die Endpunkte des Baums, an denen keine weitere Unterteilung mehr möglich ist bzw. an denen die vorgegebene Unterteilungstiefe erreicht wurde, bezeichnet man als Blätter, englisch *leaves*, oder auch als externe Knoten, *external nodes* oder als *buckets* (Bentley, 1990).

Die Datenstruktur erfüllt damit drei Funktionen auf einmal: Sie speichert den Datensatz, unterteilt den Raum in Hyperrechtecke und liefert gleichzeitig ein Verzeichnis dieser Hyperrechtecke (Bentley & Friedman, 1979).

Die Struktur des Baums und damit auch die geometrische Repräsentation sind abhängig von der Unterteilungsregel, d.h. wie, in welcher Dimension und in welcher Abfolge die Schnittebenen gebildet werden (Maneewongvatana & Mount, 1999). Man unterscheidet hier zwei Grundtypen, punktbasierte und raumbasierte K-d Trees (Goodrich & Tamassia, 2002), die auf verschiedenen Schnittregeln basieren.

Bei Punkt-basierten K-d Trees werden Unterteilungen abhängig von der Verteilung der Punkte im k-dimensionalen Raum vollzogen. Eine einfache Schnittregel legt die Reihenfolge der Schnittdimensionen als kontinuierliche Abfolge von Schnitten in x-, y- bis n-Richtung fest (Bentley, 1975; De Berg et al, 1997). Der Zyklus beginnt nach dem Schnitt in der n-ten Dimension anschließend wieder von vorne. Eine weitere mögliche Schnittregel, nach der unterteilt werden kann, ist die Standard Splitting Rule nach (Friedman et al, 1977). Hier wird die split dimension in der Dimension gewählt, in der eine Punktemenge die größte Verteilung aufweist (Goodrich & Tamassia, 2002). Der split value wird in diesem Fall aus dem Median der Punktkoordinaten berechnet.

Ein raumbasierter K-d Tree entsteht beispielsweise durch Anwendung der Midpoint Splitting Rule (Maneewongvatana & Mount, 1999). Hier wird die Schnittebene durch den Mittelpunkt der zu unterteilenden Fläche gelegt und unterteilt diese nach der längsten Seite. Eine Weiterentwicklung mit Punkt- sowie raumbasierten Eigenschaften findet sich in der von Maneewongvatana und Mount (1999) vorgestellten Sliding Midpoint Rule, bei der zwar die Region zunächst mittig unterteilt wird, die Schnittebene aber anschließend zum nächstgelegenen Punkt hin verschoben wird (Abb. 8).

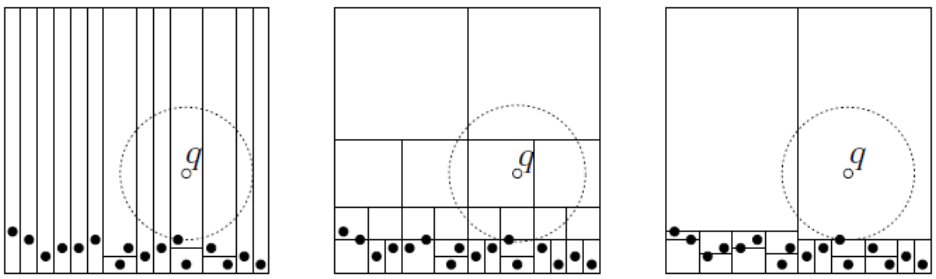


Abb. 8: Unterteilungsergebnisse mit unterschiedlichen Schnittregeln: Standard Splitting Rule, Midpoint Splitting Rule und Sliding Midpoint Rule (Maneewongvatana & Mount, 1999)

Die split values können darüber hinaus aus Median- oder Mittelwertbildung der Punktkoordinaten berechnet werden, wobei sich der strukturelle Aufbau der entstehenden K-d Trees unterscheidet. Bei der Medianberechnung wird derjenige Punkt bestimmt, dessen Koordinate in der Schnittdimension den mittigsten Wert aller Koordinaten der Punktmenge aufweist. Die Schnittebene wird anschließend durch diesen Medianpunkt geführt (Abb. 9, links). In der Folge können jedem internen Knoten des K-D Trees je ein Punkt der Punktemenge zugeordnet werden. Im Gegensatz dazu liegen alle Punkte in einem durch Mittelwertberechnung gebildeten Baum in den Blattregionen (Abb. 9, rechts). Die split values werden hier aus dem arithmetischen Mittelwert aller Punktkoordinaten in der split dimension berechnet.

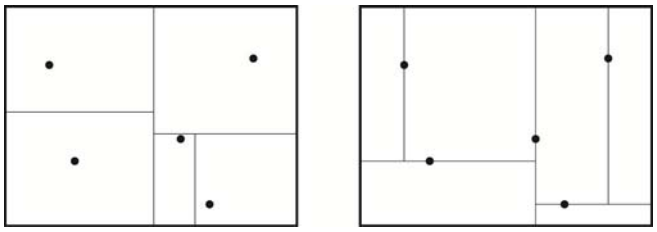


Abb. 9: Unterteilung durch Median- (links) bzw. Mittelwertberechnung (rechts)

3. Generierung von Grundrisslayouts mit K-d Trees

3.1. Generativer Mechanismus

Die Idee K-d Tree Algorithmen zur Generierung von Grundrisslayouts im Rahmen des Kremlas Forschungsprojekts einzusetzen, entstand aus einer Betrachtung ihrer geometrischen Eigenschaften sowie den Möglichkeiten, die ihre Unterteilungsstruktur und Datenstruktur bieten. In der Folge wird die Entwicklung des generativen Mechanismus beschrieben, d.h. der Aufbau der allgemeinen Datenstruktur, sowie die eingesetzten evolutionären Algorithmen zur Suche nach bestimmten Raumgrößen und Nachbarschaftsverhältnissen. Der Einsatz des K-d Trees zur Generierung von Grundrissen wurde zunächst nur auf Gebäudeebene untersucht. Die dargestellten Prototypen und Arbeitsversionen wurden in C# implementiert.

3.2. Allgemeine Datenstruktur

Die ersten Arbeitsschritte bestanden zunächst in der Umsetzung eines einfachen K-d Tree Algorithmus auf Basis der in Kapitel 2.3 dargestellten Logik, nach der eine vorgegebene Fläche durch Mittelwert- bzw. Medianberechnung in Raumachsenabfolge unterteilt wird (Abb. 10). Die Schnittlinien entsprechen dabei möglichen Raumgrenzen.

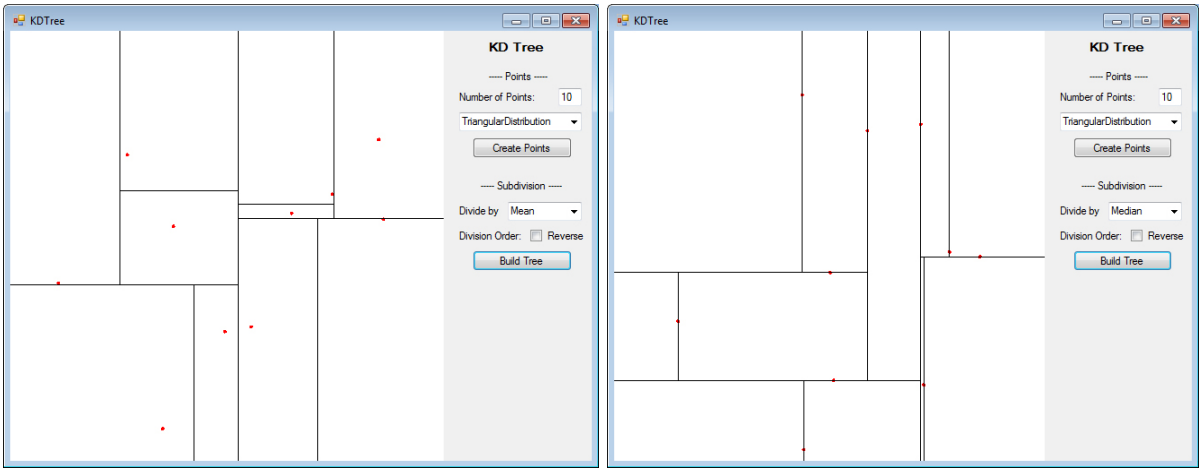


Abb.10: Erster Prototyp; Raumunterteilung durch einen einfachen K-d Tree Algorithmus mit Mittelwert- und Medianwertberechnung

Beide Berechnungsvarianten besitzen Vor- und Nachteile hinsichtlich Interaktion und Datenmanagement. Beim Aufbau der K-d Tree Datenstruktur durch Mittelwertberechnung repräsentiert jeder Punkt der Punktemenge einen Raum. Diese Repräsentationsform hat den Vorteil, dass die Größe der Punktmenge gleichzeitig die Anzahl der generierten Räume im Grundriss bestimmt und jedem Punkt folglich ein Raum zugeordnet werden kann. Erfolgt der Aufbau der K-d Tree Datenstruktur hingegen durch Medianberechnung, so geht durch jeden Punkt der Punktemenge eine Schnittlinie. Die Anzahl der entstehenden Räume berechnet sich folglich aus (Anzahl Punkte + 1).

Während durch Mittelwertberechnung entstandene K-d Trees eine logische Verknüpfung von Punkten und Räumen erlauben und damit die Zuordnung von Raumeigenschaften erleichtern, zeichnen sich durch Medianwertberechnung entstandene K-d Trees durch eine nachvollziehbarere geometrische Unterteilung und Zuordnung aus.

Die K-d Tree Datenstruktur bildete die Grundlage für alle folgenden Arbeitsschritte. Darauf aufbauend wurde zunächst ein einfacher Evolutionärer Algorithmus (Tab. 2) umgesetzt, der die mittels K-d Tree generierten Layouts im Hinblick auf die Raumgrößen optimierte (Abb. 11). Die hierzu eingesetzte evolutionäre Strategie wurde in der Folge gekoppelt mit einem genetischen Algorithmus, der die Nachbarschaftsbeziehungen der Räume optimiert. Abschließend wurde der generative Mechanismus als multikriterielles System implementiert, das die beiden Kriterienparameter so anpasst, dass sie zu optimalen Layoutlösungen führen.

Algorithmus:	Evolutionärer Algorithmus
Input:	μ – Größe der Elternpopulation λ – Größe der Kindpopulation Θr – Rekombinationsparameter Θm – Mutationsparameter Θs – Selektionsparameter
Output:	a* das beste Individuum während des Durchlaufs P' die beste Population während des Durchlaufs
Logik:	1. Generation t = 0 2. Initialisiere P(t) mit μ Individuen 3. Evaluiere alle Individuen in P(t) mit der Evaluationsfunktion F(t) 4. Rekombiniere P(t) mit der Wahrscheinlichkeit $\Theta r \rightarrow$ P'(t) 5. Mutiere P'(t) mit der Wahrscheinlichkeit $\Theta m \rightarrow$ P''(t) 6. Evaluiere alle Individuen in P''(t) mit der Evaluationsfunktion F(t) 7. Selektiere μ Individuen aus P''(t) nach F(t) und dem Selektionsparameter $\Theta s \rightarrow$ P(t+1) 8. t = t + 1 9. Beginne wieder bei Schritt 4 solange kein Abbruchkriterium erreicht ist.

Tab. 2: Evolutionärer Algorithmus nach (Bäck, 2000b)

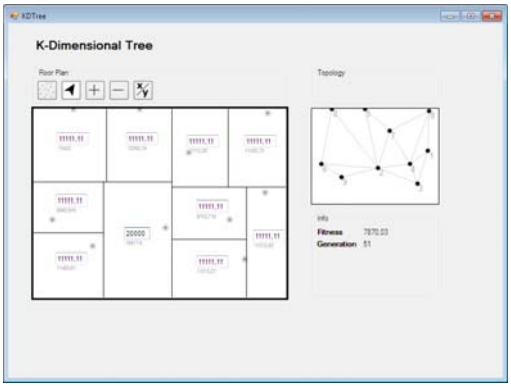


Abb. 11: Arbeitsversion zur Optimierung der Raumgrößen

3.3. Suche nach bestimmten Raumgrößen

Die Optimierung der Raumgrößen ist ein geometrisches Problem, das als Formfindungsproblem bezeichnet werden kann. Es wird im Rahmen des K-d Tree Algorithmus durch die Größe der durch Unterteilung generierten Räume bestimmt. Die Unterteilung hängt ihrerseits von der Lage und Verteilung der Punkte im Raum ab. Um die Raumgrößen zu optimie-

ren, müssen die Positionen der einzelnen Punkte so verändert werden, dass ihre Verteilung nach der Unterteilung die gewünschten Raumgrößen entstehen lässt (Abb. 12). Zur Optimierung erwies sich der Einsatz einer evolutionären Strategie als hilfreich.

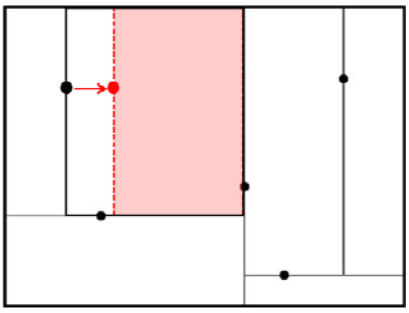


Abb. 12: Raumgrößenänderung durch Punktverschiebung

Evolutionäre Strategien zeichnen sich dadurch aus, dass die Individuen vor der Anwendung von Selektions-, Mutations- und Rekombinationsparametern nicht kodiert werden und direkt durch die evolutionären Operationen manipuliert werden.

Man unterscheidet bei Evolutionären Algorithmen verschiedene Strategien, die auf der Anzahl sowie dem Verhältnis der Eltern zur Anzahl der generierten Kinder pro Elter sowie auf dem Übertrag der Eltern in die neue Generation basieren. Bei der (μ, λ)-Evolutionsstrategie bilden μ Individuen die Elterngeneration. Aus jedem Elternindividuum werden λ Kinder erzeugt. Aus den Nachkommen werden die besten Individuen selektiert, um die neue Elterngeneration zu formen. Bei der (μ + λ)-Evolutionsstrategie werden auch die Eltern bei der Selektion der neuen Elterngeneration berücksichtigt und können, sofern sie zu den μ besten Individuen der Gesamtpopulation zählen, erneut Nachkommen erzeugen (Rechenberg, 1994).

Im Rahmen der vorliegenden Arbeit wurde in frühen Prototypen zunächst die (μ + λ)-Evolutionsstrategie eingesetzt, um gute Lösungen aus einer Elterngeneration nicht zu verlieren, sondern ihnen die Möglichkeit zu geben erneut Nachkommen zu bilden. Dies birgt jedoch grundsätzlich die Gefahr, dass eine gute Lösung und ihre Nachkommen den Lösungsraum mit Fortschreiten der Evolution dominieren und die Evolution in lokalen Maxima verharrt. Dem kann durch einen Anteil an zufällig ausgewählten, nichtoptimalen Lösungen aus der Population bei der Bildung der Elterngeneration entgegengewirkt werden.

Der evolutionäre Prozess gestaltet sich folgendermaßen. Zunächst wird eine Elterngeneration initialisiert und die Fitness der Individuen bestimmt. Die eingesetzte Bewertungsfunktion ergibt sich aus der Abweichung der tatsächlichen Flächeninhalte der Räume von ihren Idealwerten (Abb. 13):

$$f = \sum_{i=1}^{n} |A_i - A_i'|$$

wobei n = Anzahl der Flächen

A = tatsächlicher Flächeninhalt

A' = idealer Flächeninhalt

Anschließend werden durch Mutation von jedem Elternindividuum λ Kinder erzeugt. Zur Erzeugung eines Kindes werden mit einer Mutationswahrscheinlichkeit Θ_m die Positionen der Punkte mutiert, d.h. ihre Lage verändert (Abb. 12). Nach der Evaluierung aller Lösungen werden aus der Kindpopulation und ihrer Elterngeneration μ Individuen ausgewählt. Hierbei kann es sich um die μ Individuen mit den besten Fitnesswerten handeln oder, wie bereits erwähnt, um eine Kombination aus den fittesten mit einigen zufällig ausgewählten Individuen. Diese μ Individuen bilden die neue Elterngeneration.

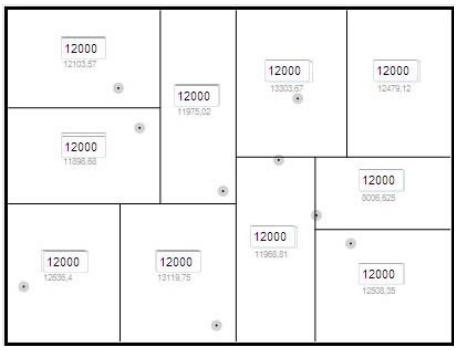


Abb. 13: Optimierung der Raumgröße von K-d Tree generierten Layouts durch eine Evolutionäre Strategie

3.4. Suche nach bestimmten Nachbarschaftsverhältnissen

Die Suche nach bestimmten Nachbarschaftsverhältnissen zwischen den Punkten bzw. den ihnen zugeordneten Räumen ist ein topologisches Problem. Es besteht darin, den Räumen die Funktionen so zuzuordnen, dass sie die gewünschten Nachbarschaftsbeziehungen aufweisen (Elezkurtaj, 2004). Darüber hinaus sind die Schnittdimensionen so zu wählen, dass durch die resultierende Unterteilung die gewünschten Lagebeziehungen zwischen Räumen entstehen. Die gewünschten Nachbarschaftsbeziehungen zwischen Räumen werden vom Nutzer definiert.

Um die Nachbarschaftsbeziehungen zu optimieren, müssen zum einen die den Flächen zugeordneten Funktionen und zum anderen die Schnittdimensionen so lange vertauscht werden bis die entstandene Layoutlösung die vorgegebene Topologie aufweist (Abb. 14).

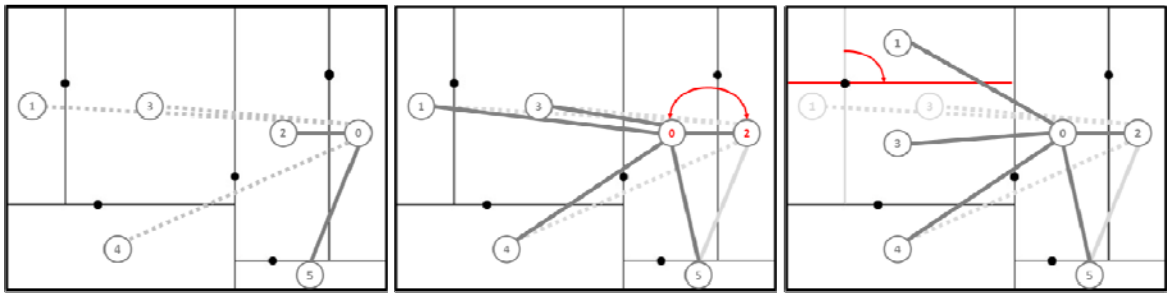


Abb. 14: Topologisches Problem (links), Optimierung der Nachbarschaften durch Vertauschen von Indizes (Mitte) und Änderung der Schnittrichtung (rechts)

Die topologische Optimierung lässt sich durch Genetische Algorithmen vornehmen. Ein grundlegender Aspekt der Genetischen Algorithmen ist, dass die Parameter der Problemstellung zunächst im Rahmen der Suche kodiert werden. Die kodierten Elemente werden als Chromosomen bezeichnet. In der ursprünglichen Variante des Genetischen Algorithmus stellen sie eine Bitfolge dar, welche die Parameter des Problems repräsentiert (Whitley, 1994). Sie sind die Genotypen, die vom Genetischen Algorithmus manipuliert werden. In abgewandelter Form kann der Genotyp auch aus einer Folge von Elementen anderer Typen bestehen (Bäck, 2000a).

Im konkreten Fall werden die Raumindizes in der Raumfolge als Chromosom kodiert und repräsentieren so den Phänotyp, eine Layoutlösung. Die Raumindizes können Werte zwischen 0 und $n\text{-}1$ annehmen, wobei n die Anzahl der Räume darstellt. Die Raumfolge ergibt sich aus der Abfolge ihrer Entstehung beim Aufbau des K-d Trees (siehe Abb. 15).

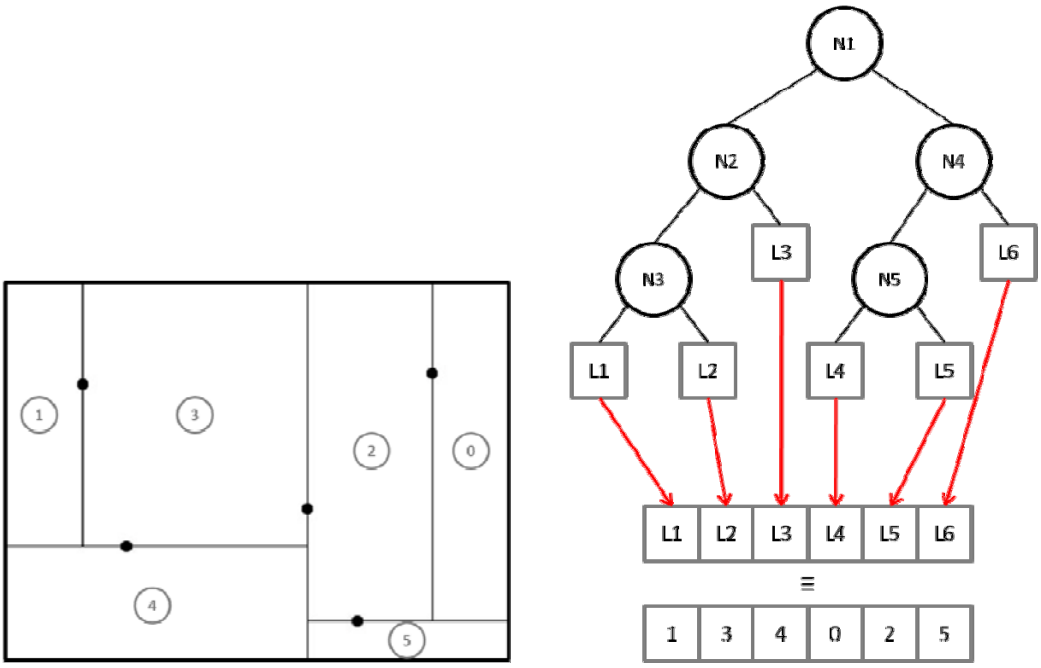


Abb. 15: Kodierung der Raumindizes aus der Raumabfolge, Phänotyp (links), Genotyp (rechts)

Die Schnittdimensionen der Layoutlösung werden als Unterteilungsfolge kodiert, die sich ebenfalls aus der Baumstruktur ergibt (Abb. 16). Anders als bei herkömmlichen K-d Trees, werden durch die Verwendung eines genetischen Algorithmus die Schnittdimensionen nicht im Voraus oder durch die Punktverteilung festgelegt, sondern können dynamisch variiert werden. Die Schnittdimensionen können Werte zwischen 0 und $k\text{-}1$ annehmen, wobei k der Anzahl der Raumdimensionen entspricht. In einem zweidimensionalen Raum wie der Layoutlösung repräsentiert 0 eine Unterteilung in x-Richtung, 1 eine Unterteilung in y-Richtung (Abb. 16).

Das Genom beschreibt die Abfolge der Schnittdimensionen in den Knoten in Reihenfolge der Baumebenen von der Wurzel bis zu den Blättern. Das Genom hat eine Länge von:

$$L = \sum_{i=0}^{n-1} 2^i$$

Wobei n die Anzahl der Baumebenen ist.

Das Genom hat damit mehr Stellen als eine Layoutlösung bzw. ein K-d Baum in der Regel tatsächliche Unterteilungen besitzt. Stattdessen besitzt es jeweils eine Stelle für jeden theoretisch möglichen Knoten auf allen Baumebenen (Abb. 16). Diese Form der Kodierung wurde gewählt, um jede Unterteilung eindeutig einer Position im Baum zuordnen zu können und umgekehrt. Würden die Unterteilungen stattdessen als kontinuierliche Folge ähnlich der Raumindizes kodiert, so könnten, bedingt durch die unterschiedliche Verteilung der Punkte und die damit in Verbindung stehende Lage der Knoten im Baum, zwei Lösungen mit demselben Unterteilungsgenom zu zwei unterschiedliche Unterteilungsstrukturen auf Ebene der Phänotypen führen.

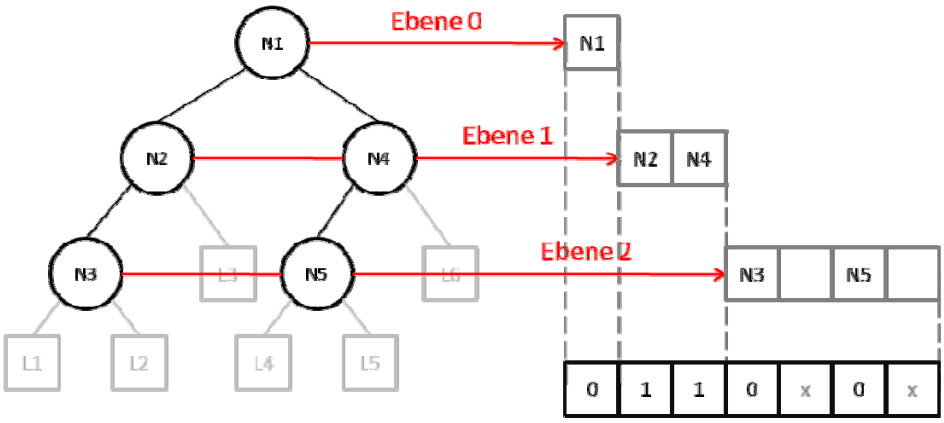


Abb. 16: Kodierung der Unterteilungsfolge

Das Genom wird immer vollständig initialisiert, so dass jederzeit Veränderungen in der Baumstruktur, d.h. Verschiebungen von Knoten und Ästen, vorgenommen werden können. Ist ein Knoten im Baum nicht besetzt, wird der zugehörige Schnittwert nicht aus dem Genom ausgelesen bzw. ist für die Bildung der aktuellen Lösung nicht relevant.

Die Evaluierungsfunktion berechnet die Summe aller Abstände zwischen den Räumen gewünschter Nachbarschaften und lässt sich folgendermaßen beschreiben:

$$f = \sum_{i=1}^{n} \overline{A_i B_i}$$

Wobei A und B die Räume darstellen, die benachbart sein sollen und n die Anzahl der Nachbarschaften.

Der Abstand zwischen zwei direkt benachbarten Räumen beträgt 0, wobei eine Mindestdurchgangsbreite bei der Überlappung gegeben sein muss, um sie als benachbart gelten zu lassen. Eine Mindestdurchgangsbreite wird berücksichtigt, um später das Platzieren von Verbindungen wie Türen zwischen den Räumen zu ermöglichen.

Der aus der Evaluierungsfunktion resultierende Fitnesswert ist umso niedriger, je mehr gewünschte Nachbarschaftsbeziehungen tatsächlich in der aktuellen Layoutlösung existieren und je näher diese Räume zueinander liegen. Ein niedriger Fitnesswert ist folglich vorteilhaft für die Performance einer Lösung. Ihr Idealwert tendiert gegen 0.

Die Selektion der Eltern zur Rekombination kann auf verschiedene Arten stattfinden, beispielsweise über Roulette-Wheel-Selection oder wie die in diesem Fall verwendete Binary Tournament Selection. Hier werden zwei Individuen der Elterngeneration zufällig ausgewählt, wobei das Individuum mit dem besseren Fitnesswert zur Reproduktion bestimmt wird.

Die Rekombination zweier Elternindividuen wird mittels One-Point-Crossover durchgeführt, bei dem die Chromosomen der Eltern an einer Schnittstelle getauscht werden, so dass zwei neue Chromosomen entstehen (Abb. 17).

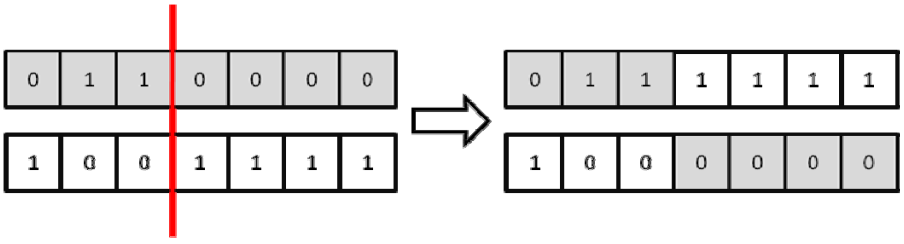


Abb. 17: One-Point-Crossover

Mit der Wahrscheinlichkeit Θ_m werden diese Kinder anschließend mutiert. Mutation bedeutet in diesem Fall, dass die Indizes einzelner Räume miteinander vertauscht werden bzw. die Dimension einer Stelle im Genom geändert wird. Durch (μ + λ)-Selektion wird eine neue Elterngeneration gebildet.

Die Suche nach bestimmten Nachbarschaftsverhältnissen erfolgte zunächst, ebenso wie die Suche nach bestimmten Raumgrößen, in einer eigenständigen Population. Die Nachbarschaftsbeziehungen bzw. die Raumgrößen wurden folglich in zwei Populationen parallel optimiert. Zusätzlich wurden die besten Individuen bzw. die besten Lösungen in jeder Generation zwischen den Populationen ausgetauscht, d.h. migriert (Abb. 18). Diese erste Optimierungsstrategie wurde in der Folge durch ein multikriterielles System ersetzt, da die Migration die Generierung von in beiden Kriteriendimensionen optimalen Lösungen nur in geringem Maße beeinflusste bzw. begünstigte.

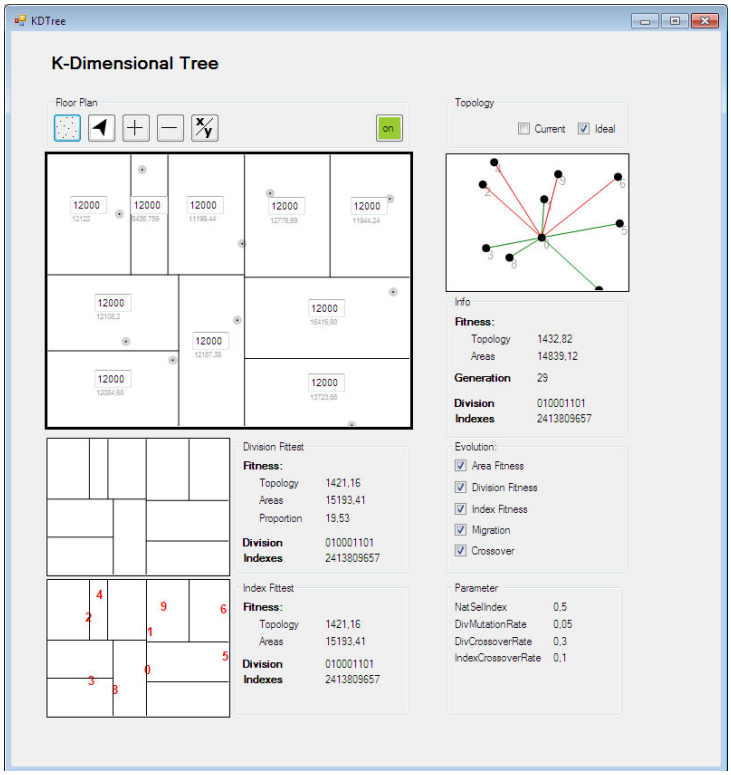


Abb. 18: Optimierung in zwei Kriteriendimensionen, nach Raumgrößen und Nachbarschaften

3.5. Multikriterielles System

Wie gesehen beinhalten Layoutprobleme oft konkurrierende Parametersets. Die Lösung solcher Probleme besteht darin, die Parameterwerte so anzupassen, dass sie eine optimale Lösung ergeben. Im Gegensatz zu Problemstellungen mit nur einem Optimierungsziel bedeutet dies, das bei der Lösung Kompromisse eingegangen werden müssen, da häufig die Verbesserung einer Lösung hinsichtlich eines Parameters die Verschlechterung des Ergeb-

nisses hinsichtlich eines anderen Parameters bedeutet. Man spricht hier von Multikriteriellen Optimierungsproblemen (MOPs) (Zitzler, 1999).

Multiobjective Optimization wird definiert als ein Problem zur Findung eines *„vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions."* (Coello Coello, 1998, S. 2).

Im Allgemeinen können, wie bei Horn (1997) beschrieben, zwei Problemstellungen bei der Lösung von MOPs ausgemacht werden: Die Suche nach geeigneten Lösungen in einem komplexen Lösungsraum und die multikriterielle Entscheidungsbildung, d.h. die Auswahl einer geeigneten Kompromisslösung aus einem Set an Lösungen, die von einem menschlichen Entscheidungsträger getroffen werden muss.

Abhängig von der Kombination von Suche und Entscheidungsfindung im Optimierungsprozess können die eingesetzten Optimierungsmethoden unterschieden werden in solche, bei denen die Entscheidungsbildung vor, nach oder während der Suche stattfinden (Horn, 1997). In der ersten Kategorie werden die Kriterien vor der Suche durch Präferenzbildung in ein einzelnes Kriterium überführt, wie beispielsweise im Weighted Sum Approach (Cohon, 1978). Dadurch können die Optimierungsprobleme wie einzelkriterielle Probleme und mit traditionellen Methoden gelöst werden. Dieser Ansatz setzt jedoch bereits ein erhebliches Vorwissen über die Gewichtung der verschiedenen Parameter für die Lösungssuche voraus, die nicht bei allen Problemstellungen gegeben ist (Zitzler, 1999).

Methoden der zweiten Kategorie führen zunächst die Optimierung ohne Präferenzen durch. Es ergibt sich ein Set an Lösungen im Entscheidungsraum, pareto-optimales Set oder auch Pareto-Front genannt, aus dem der Entscheidungsträger auswählt (Zitzler et al, 2003). Lösungen im pareto-optimalen Set gelten dann als pareto-optimal, wenn eine Lösung von keiner anderen Lösung dominiert wird, d.h. wenn ihnen unter Betrachtung aller Kriterien keine andere Lösung überlegen ist (Zitzler, 1999). Sie können in keinem Kriterium verbessert werden, ohne dass sie sich nicht in mindestens einem anderen Kriterium verschlechtern.

In der dritten Kategorie kann der Entscheidungsträger bereits während der Suche Präferenzen treffen. Er erhält nach jedem Suchlauf eine Auswahl an Alternativlösungen aufgrund derer er weitere Kriterien bestimmt, die in den weiteren Suchlauf mit einfließen. Dieser sowie der zuvor beschriebene Ansatz erhöhen die Komplexität des Suchraums dadurch, dass zunächst keine Eingrenzung getroffen wird. Eine Schwierigkeit besteht insbesondere in höher-dimensionalen MOPs in der Darstellung der Ergebnisse (Zitzler, 1999).

Da die Suche nach dem pareto-optimalen Set rechenintensiv und aufgrund der Komplexität nicht direkt berechnet werden kann, wurden für die Lösungsprozesse in der zweiten und dritten Kategorie stochastische Suchstrategien entwickelt, die sich der Pareto-Front langsam annähern. Dazu zählen neben der Ant Colony Optimization, dem Simulated Anealing oder der Tabu Search insbesondere auch die Evolutionäre Algorithmen, da man mit ihnen zum einen große Suchräume sowie komplexe Problemstellungen handhaben und zum anderen durch ihre auf Populationen basierende Struktur innerhalb eines Evolutionslaufs mehrere pareto-optimale Lösungen finden kann (Zitzler et al, 2003).

Im Bereich der sogenannten Multiobjective Evolutionary Algorithms (MOEAs) existieren verschiedene Ansätze. Sie unterscheiden sich im Allgemeinen in der verwendeten Selektionsstrategie, d.h. wie die Fitness der Individuen bestimmt wird und diese zur Mutation und Bildung der Population ausgewählt werden. MOEAs wie der Vector-Evaluated Genetic Algorithm (VEGA) optimieren beispielsweise einzelne Kritierien parallel. Andere, wie der Multiobjective Genetic Algorithm (MOGA) gewichten Kriterien und verwenden eine summierte Fitnessfunktion. Darüber hinaus kann sich der Optimierungsprozess an der Pareto Front orientieren, wie zum Beispiel bei Pareto-Ranking (Goldberg, 1989), bei dem die Individuen einer Population nach deren Pareto-Optimalität sortiert werden. Weitere Selektionskriterien bieten das Niching, das die Fitness bezogen auf die Umgebung eines Individuums im Lösungsraum berechnet, und verschiedene Arten der Tournament Selection, in denen eine Gruppe an Individuen ausgewählt und verglichen werden um das fitteste Individuum zur Selektion zu bestimmen (Horn, 1997).Viele MOEAs basieren auf einer Kombination der verschiedenen dargestellten Möglichkeiten und variieren oder erweitern deren Grundkonzepte.

Im Rahmen des in Kapitel 3.1. vorgestellten Generativen Mechanismus handelte es sich bei den zu optimierenden Parametern um die Raumgrößen und die Nachbarschaftsbeziehungen in architektonischen Layouts. Die Schwierigkeit besteht darin, diese unterschiedlichen Parameter so zu kombinieren, dass Lösungen mit möglichst optimalen Raumgrößen und Nachbarschaften entstehen können. Dazu wurden in einer ersten Versuchsreihe die Kriterien zunächst auf Basis des VEGA-Ansatzes in jeweils eigenen Populationen parallel optimiert. Die Individuen dieser Subpopulationen wurden zu einer gemeinsamen Gesamtpopulation vermischt (schematische Darstellung in Abb. 21). Zur Generierung der Kindgeneration wurden die in 3.1. beschriebenen evolutionären Operationen verwendet. Aus den entstandenen Kindern wurden anschließend die neuen Subpopulationen erstellt.

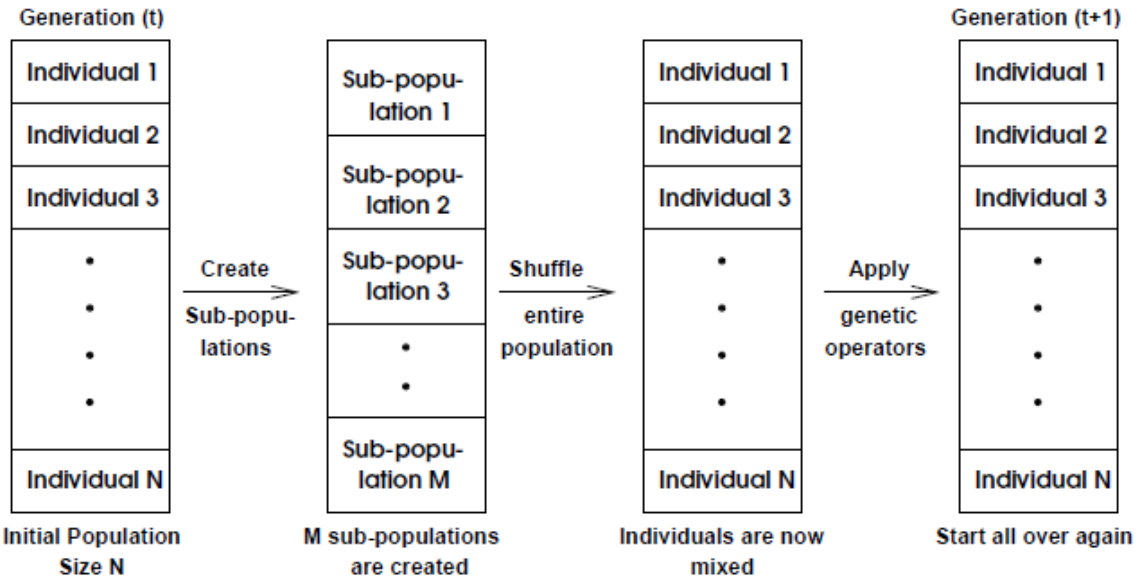


Abb. 19: Schematische Darstellung des VEGA, wobei N die Größe der Gesamtpopulation und M die Anzahl der Kriterienfunktionen darstellt (entnommen aus Coello Coello, 1998)

Der VEGA zeichnet sich vor allem durch seine Einfachheit aus. Aus seinem Aufbau ergeben sich jedoch auch Probleme. Zum einen ist ein lokal in einer Subpopulation nicht-dominiertes Individuum nicht notwendigerweise auch global nicht-dominiert, d.h. lokal optimale Individuen sind nicht notwendigerweise auch global besser als andere. Zum anderen unterstützt der Selektionsprozess die Bildung von Spezies, also Gruppen von Individuen, die in verschiedenen Kriterien besonders gute Ergebnisse liefern, da vor allem diese Individuen selektiert werden. Dadurch gehen solche Lösungen verloren, die eine mittelmäßige Performance in allen Kriteriendimensionen erreichen und die zur Auswahl von Kompromisslösungen günstig wären (Coello Coello, 1998).

Aufgrund der genannten Nachteile von VEGA wurde der Pareto Envelope-based Selection Algorithm (PESA) getestet, der bei der Selektion und Optimierung die Lösungsvielfalt über den Lösungsraum hinweg erhält und die Pareto Front speziell in noch unerforschte Bereiche vorantreibt (Corne et al, 2000).

PESA zeichnet sich durch eine kleine interne Population aus, welche die neugebildeten Individuen enthält, die noch evaluiert werden müssen, und verfügt über eine meist größere externe Population, auch als Archiv bezeichnet, das die Individuen enthält, welche die aktuelle Pareto Front bilden. Darüber hinaus wird der Lösungsraum gerastert, sodass ein sogenanntes hyper grid entsteht, anhand dessen die Verteilung bzw. die Häufung der Lösungen auf der Pareto Front kontrolliert wird. Dieser Häufungswert, der sogenannte Squeezefaktor, dient als Auswahlkriterium für die Individuen (Corne et al, 2000).

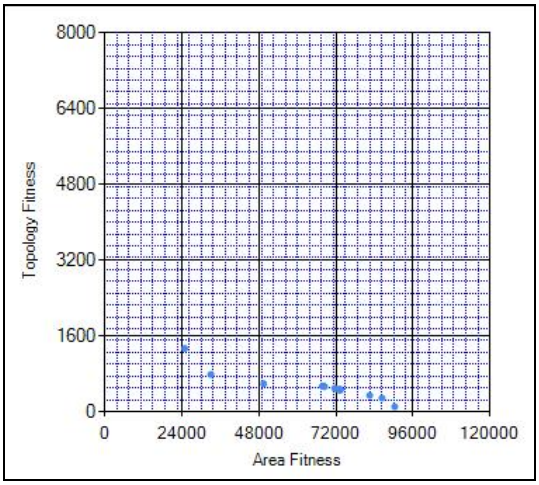
Algorithmus:	Pareto Envelope-based Selection Algorithm (PESA)
Parameter:	IP = Interne Population P_I = Anzahl der Individuen in der internen Population $P_{E,max}$ = Maximale Anzahl an Individuen in der Externen Population p_c = Crossover Wahrscheinlichkeit
Output:	EP = Externe Population, Pareto-Front von optimalen Lösungen
Logik:	1. Generation **t = 0** 2. Initialisiere **IP(t)** mit P_I Individuen und initialisiere eine leere **EP(t)** 3. Evaluiere **IP(t)** und verschiebe nicht-dominierte Elemente von **IP(t)** nach **EP(t)** 4. Wenn die maximal zugelassene Anzahl an Individuen $P_{E,max}$ in **EP(t)** überschritten ist, lösche Individuen aus **EP(t)** bis Anzahl gleich $P_{E,max}$ 5. Wenn ein Abbruchkriterium erreicht wurde, halte den Algorithmus an und gebe das Set an Individuen in **EP(t)** als Ergebnis zurück Sonst leere **IP(t)** und führe die folgenden Anweisungen aus bis P_I neue Lösungen generiert wurden: a. Wähle zwei Eltern mit einer Wahrscheinlichkeit p_c aus **EP(t)** aus, produziere ein Kind durch Crossover und mutiere das Kind b. Wähle mit der Wahrscheinlichkeit **(1 – p_c)** ein Elternindividuum aus **EP(t)** aus und mutiere es, um ein Kind zu erstellen 6. t = t + 1 7. Beginne wieder bei Schritt 3

Tab. 3: Pareto Envelope-based Selection Algorithm (Corne et al, 2000)

Der PESA Algorithmus ist in Tabelle 3 schematisch dargestellt. Die Rekombination und Mutation aus Schritt 5 erfolgten analog zu den in den Kapiteln 3.3 und 3.4 beschriebenen Schemata. Die Dominanz zwischen den Individuen wurde über die ebenfalls dort beschriebenen Bewertungsfunktionen bestimmt. Bei der Verschiebung der nicht-dominierten Elemente aus der internen in die externe Population wird Individuum für Individuum geprüft, ob es von keiner anderen Lösung in der internen und der externen Population dominiert ist. Nur wenn es nicht dominiert ist wird es ins Archiv übernommen. Gleichzeitig werden auch alle Archivlösungen neu evaluiert und dominierte Individuen gegebenenfalls aus der externen Population entfernt. Überschreitet die Anzahl an Individuen in der Externen Population eine festgelegte Höchstzahl, werden Individuen auf Basis des Squeezefaktors entfernt (Corne et al, 2000).

Der Squeezefaktor berechnet sich folgendermaßen: Der mehrdimensionale Lösungsraum wird durch ein Hyperraster in Hyperboxen unterteilt, wobei jedes Individuum einer Hyper-

box zugeordnet werden kann. Die Anzahl der Lösungen in einer Box bestimmen deren Squeezefaktor. Für das Update des Archivs wird zunächst der maximale Squeezefaktor der Population bestimmt und anschließend ein Element aus der Hyperbox mit dem größten Squeezefaktor zufällig ausgewählt und entfernt.



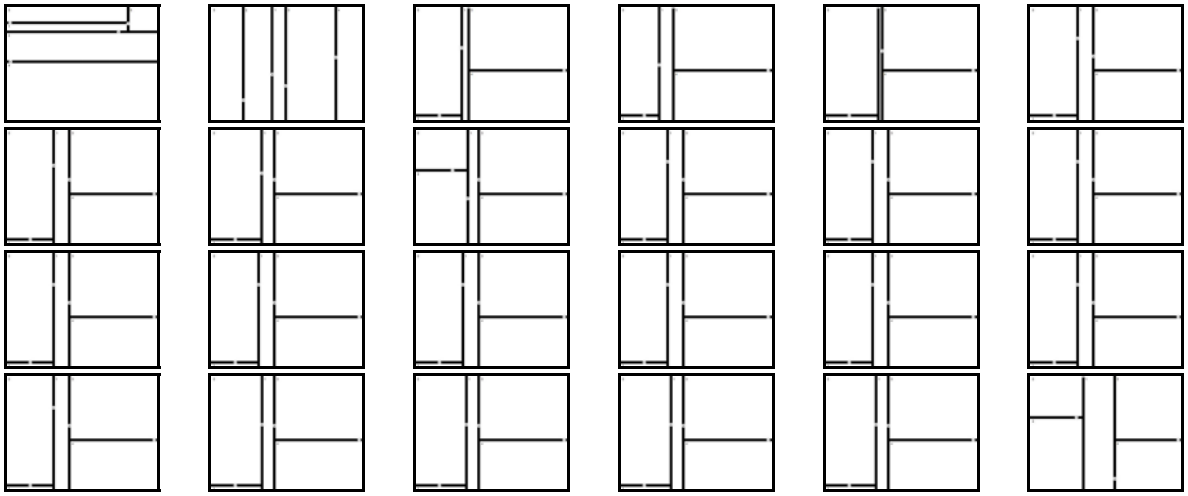


Abb. 20 und 21: Pareto-Front (oben) und zugehörige Archivlösungen (unten) (Generation: 9)

Die Auswahl der Eltern erfolgt auf Basis der Häufung der Archivlösungen in bestimmten Regionen durch Binary Tournament Selection. Zwei Lösungen werden hierbei durch Zufall aus dem Archiv ausgewählt. Das Individuum mit dem kleinsten Squeezefaktor wird zur Generierung des Kinds hergenommen, um die Pareto Front in weniger frequentierte Regionen zu erschließen.

Durch den PESA entwickelte sich eine Pareto-Front, die relativ schnell gute Ergebnisse in der Optimierung der Nachbarschaftsbeziehungen und der Raumgrößen zeigte.

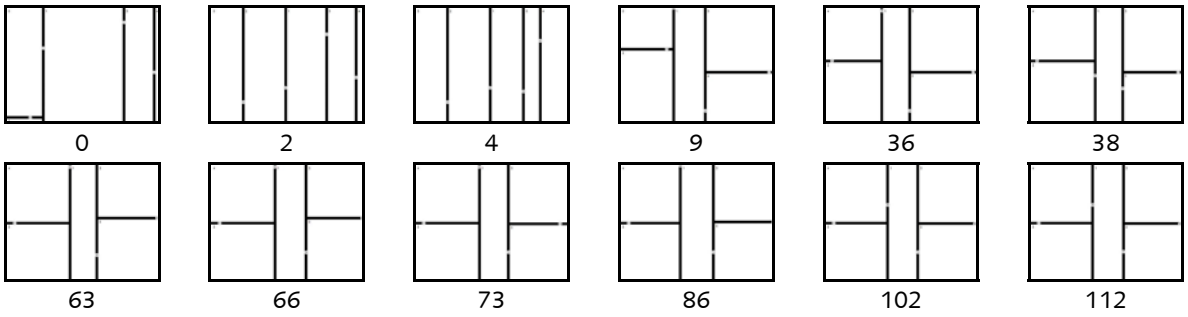


Abb. 22: Optimierung der Raumgrößen: Fitteste Individuen nach Generationen 0 bis 112

3.6. Verschachtelte K-d Trees

Die Datenstruktur des K-d Trees unterstützt darüber hinaus das Konzept der in Kapitel 2.2. erwähnten verschachtelten Elemente. Layoutelemente entsprechen externen Knoten, also Elementen untergeordneter Hierarchieebene, denen anschließend bestimmte Funktionen innerhalb des Layouts zugeordnet werden können. Interne Knoten entsprechen Elementen übergeordneter Hierarchieebenen, die untergeordnete interne und externe Knoten beispielsweise zu Funktionsbereichen bündeln können. Die internen und externen Knoten stellen in der Implementierung folglich zentrale Objekte dar, die durch einen Datenpunkt und seine Lage sowie die zugehörige Region definiert werden. Gleichzeitig kann auch der K-d Tree selbst als Objekt betrachtet werden, das die Baumstruktur als Referenzen zwischen Root und Child Nodes speichert.

Im Rahmen einiger Prototypen wurde das Prinzip der verschachtelten K-d Trees versuchsweise implementiert (Abb. 19 und 20). Dazu wurde zunächst die Grunddatenstruktur des K-d Trees angepasst und zusätzliche Verweise zwischen parent und child trees eingefügt, die es ermöglichten in den leaves eines K-d Trees ein diesem Baum untergeordnetes child tree zu pflanzen. Die dem leaf zugeordnete Region gibt die Begrenzungen für dieses child tree vor.

Jeder verschachtelte K-d Tree stellt eine Layoutproblematik für sich dar, die es im Rahmen des generativen Mechanismus zu lösen gilt. Folglich ist auch der generative Mechanismus verschachtelt. Jede Layoutanpassung und –veränderung durch evolutionäre Prozesse oder Nutzerinteraktion auf einer höheren Hierarchiestufe bedingt automatisch die Anpassung und Veränderung der abhängigen, untergeordneten Strukturen und ihrer evolutionären Prozesse.

Theoretisch ist es denkbar K-d Trees unendlich tief zu verschachteln, praktisch stößt die Verschachtelung allerdings an Grenzen. Diese Grenzen betreffen die Umsetzbarkeit. Beispielsweise kommt es mit zunehmender Verschachtelungstiefe zu Darstellungs- und Re-

chenleistungsproblemen, oder die Tiefe der Verschachtelung ist begrenzt durch die Existenz von konkreten, sinnvollen Entsprechungen auf architektonischer und städtebaulicher Ebene.

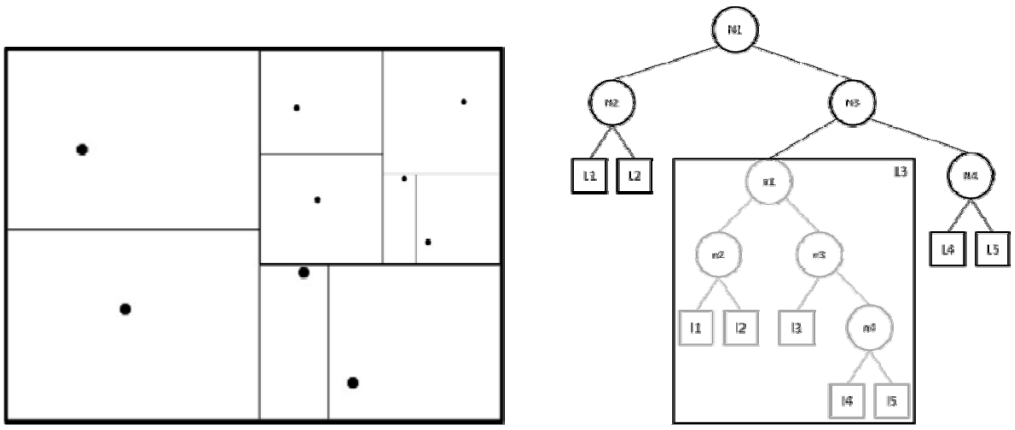


Abb. 23: Geometrische und grafische Darstellung eines verschachtelten K-d Trees

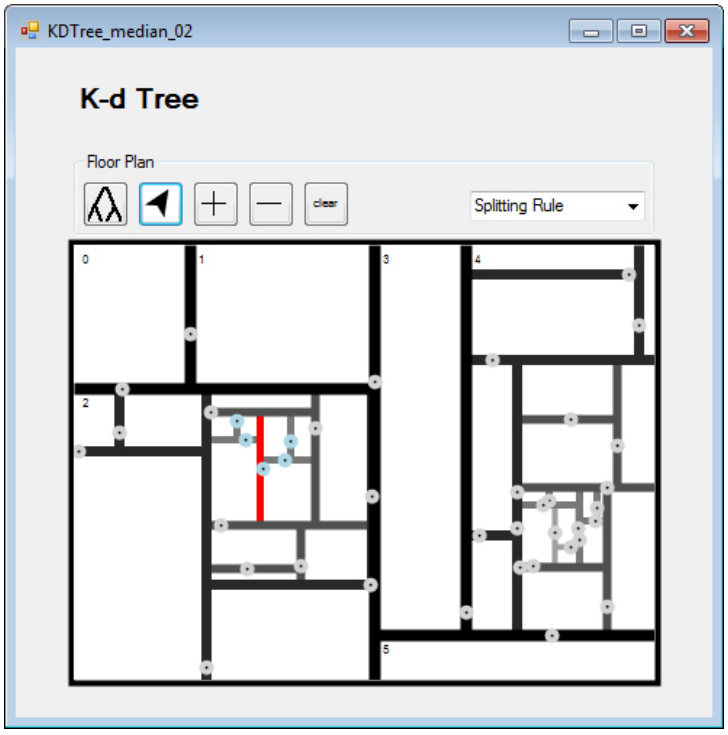


Abb. 24: Prototypische Umsetzung eines verschachtelten K-d Trees

3.7. Nutzerinteraktion

Da architektonische Fragestellungen auch nicht-operationale, wie beispielsweise ästhetische Kriterien behandeln, die nicht eindeutig beschrieben und damit nicht mithilfe von Algorithmen optimiert werden können, muss ein generatives System eine Möglichkeit bzw. eine Schnittstelle zur Berücksichtigung nicht-operationaler Kriterien bieten. Dies kann durch direktes Eingreifen des Nutzers in den Lösungsfindungsprozess durch ein User Inter-

face geschehen, das die Interaktion mit den optimierten Layoutlösungen ermöglicht. Aus dem Set an Lösungen in der Pareto Front, die aus der wie in Kapitel 3.6. beschriebenen Optimierung nach den operationalen Kriterien der Raumgrößen und Nachbarschaften entstanden, wählt der Entscheidungsträger nach nicht-operationalen Kriterien, wie ästhetischen Vorzügen, die gewünschte Lösung aus (Zitzler et al, 2003) bzw. passt sie an.

In den entstandenen Prototypen erfolgt der Eingriff in den Optimierungsprozess direkt anhand der grafischen Repräsentation der Lösung durch Mausklick, Mausbewegung oder Tastatureingabe. Da der generative Mechanismus auf K-d Trees basiert, umfassen die Interaktionsmöglichkeiten in der prototypischen Anwendung zunächst vor allem Funktionen zur Manipulation der Punkte. Zu diesen Möglichkeiten gehören das Hinzufügen und Löschen sowie das Verschieben von Punkten, durch die der Nutzer auf die Anzahl der Räume sowie deren Verteilung und Gestalt Einfluss nehmen kann.

Die Interaktion des Nutzers mit dem Layout gestaltet sich bei Mittelwert- und Medianberechneten Bäumen unterschiedlich (Abb. 25). Bei mittelwertberechneten Layoutlösungen dienen insbesondere die in den Räumen liegenden und mit ihnen verknüpften Punkte als Interaktionselemente. Durch Verschieben der Punkte lassen sich die Lage und Größe der Räume verändern (Abb. 25, links). Da die Anpassung jedoch immer gemittelt wird, d.h. die Verschiebung des Punktes um eine Strecke s nicht gleichzeitig den Raum oder die Raumgrenzen um s verschiebt, ist sie für den Nutzer nicht immer nachvollziehbar und die Interaktion zunächst gewöhnungsbedürftig. Die Anpassung kann demzufolge nur ungenau erfolgen.

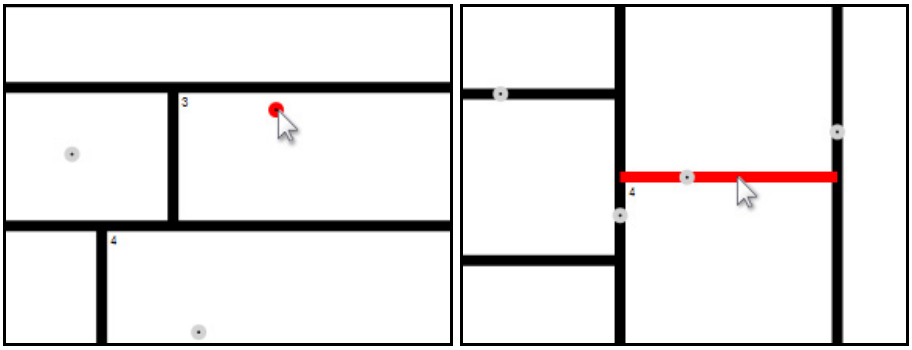


Abb. 25: Punkt- (links) und linienbasierte (rechts) Interaktion

Bei medianberechneten Bäumen dienen insbesondere die Schnittlinien, d.h. die Raumbegrenzungen auf denen die Punkte liegen, als Interaktionselemente (Abb. 25, rechts). Durch Verschiebung der Linien oder der zugehörigen Punkte kann der Nutzer die Räume und Größen beeinflussen, wobei die Verschiebung der Linie um die Strecke s die Raumgrenze auch grafisch um s verschiebt. Im Gegensatz zur Interaktion bei mittelwertberechneten

Bäumen ist die Reaktion des Systems in diesem Fall leicht verständlich und lässt eine relativ genaue Anpassung zu. Probleme ergeben sich aus der auf die Schnittlinien beschränkten Interaktion. Da lange Schnittlinien nicht unterteilt werden können, verändern sich bei ihrem Verschieben die Raumgrenzen mehrerer Räume gleichzeitig.

Das Verschieben der Schnittlinien als Interaktionsform wurde auch im Zusammenhang mit mittelwertberechneten Layoutlösungen untersucht. Das Verschieben einer Linie führt hier zu einer Verschiebung der Punkte aller angrenzenden bzw. in der Baumhierarchie betroffenen Räume und ist damit grundsätzlich möglich. Es bietet eine Möglichkeit auch bei mittelwertberechneten Layouts genaue räumliche Anpassungen vorzunehmen.

Auch Hinzufügen und Löschen von Punkten hat in beiden Varianten verschiedene Auswirkungen. Bei mittelwertberechneten Layouts wird ein Raum hinzugefügt bzw. gelöscht, bei medianberechneten wird eine Unterteilung hinzugefügt bzw. weggenommen. Für beide Varianten gilt, dass die K-d Tree Struktur nicht vom Nutzer beeinflusst werden kann. Bei mittelwertberechneten Bäumen führt das Hinzufügen oder Wegnehmen eines Punkts zur Neuberechnung der Unterteilungen und damit einer Neuaufteilung der Räume. Dies gilt in geringerem Maße ebenfalls für medianberechnete Bäume. Da bei medianberechneten Bäumen die Lage der Schnittlinien durch die Knotenpunkte bestimmt wird, ändern sich beim Hinzufügen oder Wegnehmen von Punkten die Lagen der Schnittlinien nicht. In Abhängigkeit von der Schnittfolge kann sich jedoch die Schnittdimension in betroffenen Knotenpunkten ändern, d.h. die Unterteilung von horizontal zu vertikal oder umgekehrt wechseln, und zu einem neuen Raumlayout führen.

Die topologische Struktur der Layoutlösung (Abb. 26) wird auf dem Nutzerinterface in einem eigenen Bereich angezeigt. In diesem Bereich kann der Nutzer die gewünschte Topologie anpassen, d.h. die Nachbarschaftsbeziehungen zwischen den Raumfunktionen neu definieren. Die Werte für gewünschte Raumgrößen kann der Nutzer durch Tastatureingabe verändern (Abb. 27).

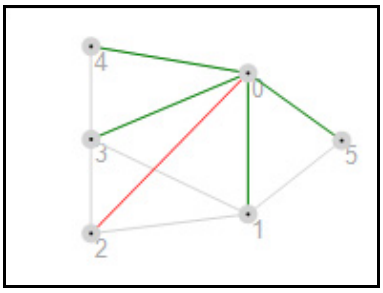


Abb. 26: Topologische Struktur einer Layoutlösung

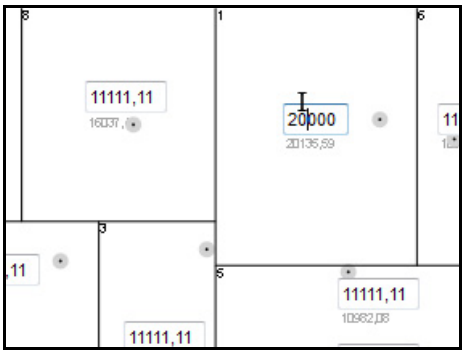


Abb. 27: Eingabe gewünschter Raumgrößen über die Tastatur

Die Eingaben und Aktionen des Nutzers haben die unmittelbare Anpassung der Populationen im Hinblick auf die veränderten Objekteigenschaften zur Folge. Da der evolutionäre Prozess kontinuierlich im Hintergrund weiterläuft, werden neue Lösungen nahezu zeitgleich ausgegeben und erlauben wiederum eine unmittelbare Neuanpassung durch den Anwender.

Die parallel laufende Optimierung der Lösung stellt die Interaktion jedoch auch vor Probleme. Aus Gründen der Übersichtlichkeit wird im aktuellen Prototyp nur eine Lösung der Pareto-Front angezeigt. Dabei handelt es sich um die Lösung, deren Flächen die optimalste Größenverteilung aufweisen. Ändert sich die Pareto-Front, so wechselt auch die angezeigte Lösung. Da die Lösungen stark voneinander abweichen können, geschieht dieser Wechsel in der Regel sprunghaft. Diese Änderung ist zwar für den Nutzer nachvollziehbar, es beeinträchtigt aber seine Interaktion mit der Lösung.

Ein weiteres Problem besteht darin, dass nicht alle Änderungen, die der Nutzer in einer Lösung vornimmt an die anderen Individuen der Population weitergegeben werden können. Dies gilt insbesondere für Änderungen die in Bezug auf die angezeigte Lösung erfolgen und damit nur lokalen Bezug haben, wie beispielsweise Verschiebungen. Die Anzahl der Räume, die idealen Raumgrößen und die idealen Nachbarschaftsbeziehungen sind hingegen globale Änderungen, die auf die gesamte Population übertragen werden können.

Die lokalen Änderungen, die der Nutzer durchführt, können darüber hinaus eine Lösung hinsichtlich eines Kriteriums verschlechtern und dazu führen, dass eine Lösung aus der Pareto-Front verschwindet bzw. die Änderung durch den Optimierungsprozess innerhalb kürzester Zeit aufgehoben wird. Dies führt zu Irritationen und beeinträchtigt den kreativen Gestaltungsprozess.

Es stellt sich folglich das Problem eine sinnvolle Balance zwischen multikriterieller Optimierung und Nutzereingabe zu finden, welches im Hinblick auf das Interaktionskonzept des in

Kapitel 2.2. beschriebenen Gesamtsystems betrachtet und gelöst werden muss. Aus den genannten weiteren Problemstellungen ergeben sich zudem die folgenden, weiteren im Kontext des Gesamtsystems zu betrachtende Fragestellungen: Wie können die Lösungen der Pareto-Front angezeigt und wie kann die Interaktion des Betrachters mit diesen Lösungen sinnvoll gestaltet werden.

4. Schlussbetrachtung und Ausblick

Im Rahmen der vorgestellten Arbeit wurden die geometrischen Qualitäten und gestalterischen Möglichkeiten der Raumunterteilung mittels K-d Tree Algorithmen zur Lösung von Layoutproblemen in Architektur und Städtebau ausgelotet. Es wurde zunächst ihre Einsatzmöglichkeit als generativer Mechanismus auf Gebäudeebene zur Generierung von Grundrissen betrachtet. In den entsprechenden Untersuchungen hat sich gezeigt, dass K-d Tree Algorithmen aufgrund ihrer geometrischen Struktur in Verbindung mit evolutionären Strategien und genetischen Algorithmen interessante Layoutlösungen für Grundrisse generieren können, deren Weiterverwertung jedoch noch untersucht werden muss. Durch einen flexiblen Aufbau der K-d Tree Struktur ohne festgelegte Teilungsregel und den Einsatz median- und mittelwertbezogener Berechnungen zeigen sich die entstandenen Layout-Lösungen flexibel, divers und können dynamisch angepasst werden.

Die eingesetzten evolutionären Algorithmen haben positiven Einfluss auf die geometrischen Eigenschaften der K-d Trees. Sie können Größen und Nachbarschaften der Unterräume steuern helfen und damit die Gestalt der Gesamtstruktur beeinflussen. Die punktbasierte Charakteristik des K-d Tree Algorithmus bietet dabei Vorteile bei der Generierung der Lösungen sowie deren Optimierung. Durch den Aufbau eines multikriteriellen Optimierungssystems können Zielkriterien über den gesamten Lösungsraum hinweg optimiert werden. Die entstehende Pareto-Front aus nicht dominierten Lösungen kann dem Nutzer bei der Entscheidungsfindung und der Auswahl von Kompromisslösungen unterstützen. Im Idealfall soll es dem Anwender möglich sein, den Lösungsraum flexibel zu durchsuchen und in seinem Sinne zu verändern. Hierzu muss das bisher entwickelte System allerdings noch erweitert werden, um dem Nutzer sinnvolle Funktionen und Interaktionsmöglichkeiten zur Verfügung stellen zu können.

Die punktbasierte Arbeitsweise der K-d Trees und die damit verbundenen, eingeschränkten Möglichkeiten der Layout-Manipulation erschweren jedoch, wie in Kapitel 3.7 dargestellt, eine intuitive, umfassende Nutzerinteraktion. Neben einer Anpassung des Interaktionskonzepts bestünden nächste Schritte darüber hinaus in der Untersuchung der Maßstabsüber-

greifenden Anwendung des Partitionierungsalgorithmus sowie der Verbindung von Layoutelementen über Layout- und Maßstabsebenen hinweg. Hier müsste überprüft werden, inwieweit die verwendete K-d Tree Struktur insbesondere die Ausgestaltung von Elementen mit festgelegten Eigenschaften unterstützt, die z.B. für Funktionseinheiten wie Treppenhäusern und andere Erschließungs- und Versorgungstrakten nötig ist. Bisher bedingt beispielsweise die Verschiebung eines Punktes in einer mittelwertberechneten K-d Tree Struktur gleichzeitig die dynamische Anpassung von Größe und Proportion benachbarter Elemente, wodurch feste Elementeigenschaften schwer aufrecht zu erhalten sind. Im Rahmen dieser Untersuchungen dürfte zudem eine konsistente Benutzerführung nicht außer Acht gelassen werden, die einen nahtlosen Übergang zwischen den verschiedenen Maßstabsebenen ermöglicht.

Im Rahmen der weiteren Ausarbeitung des vorgestellten Systems wären Erweiterungen des multikriteriellen Optimierungsmodells sinnvoll. Zu untersuchen wären zum Beispiel die Erweiterung um topologische Kriterien wie die Raumausrichtung, oder die Integration von klassischen Analysewerkzeugen für Sonnenstand und Sichtbarkeiten.

Darüber hinaus erschließen die umgesetzten generativen Mechanismen zum aktuellen Zeitpunkt nur einen begrenzten Teil des möglichen Lösungsraums, da zunächst eine Beschränkung auf orthogonale Unterteilungen und rechteckige Raumstrukturen stattfand. Im Rahmen einer Weiterbearbeitung bestünde die Möglichkeit, die Generierung von komplexen Geometrien zu untersuchen und damit die orthogonale Unterteilungsweise der K-d Trees auf nichtrechtwinklige Ebenen zu erweitern.

Abschließend kann festgestellt werden, dass die Raumpartitionierung durch K-d Trees in Verbindung mit evolutionären Algorithmen bei der Entwicklung von Methoden zur kreativen algorithmischen Lösung von Layoutaufgaben in Architektur und Städtebau eine interessante und vielversprechende Variante zu bereits bekannten Strategien darstellt.

5. Referenzen

Anders, F., & König, R. (Juni 2011). Analyse und Generierung von Straßennetzwerken mittels graphenbasierter Methoden. *Arbeitspapiere/Working Papers* . Weimar: Bauhaus-Universität Weimar, Professur Informatik in der Architektur.

Arvin, S. A., & House, D. H. (2002). Modeling architectural design objectives in physically based space planning. *Automation in Construction 11* , S. 213-225.

Bäck, T. (2000a). Binary Strings. In T. Bäck, D. B. Fogel, & Z. Michalewicz, *Evolutionary Computation 1 - Basic Algorithms and Operators* (S. 132-135). New York, NY: Taylor & Francis Group, LLC.

Bäck, T. (2000b). Introduction to Evolutionary Algorithms. In T. Bäck, D. B. Fogel, & Z. Michalewicz, *Evolutionary Computation 1 Basic Algorithms and Operators* (S. 59-63). New York, NY: Taylor & Francis Group, LLC.

Bentley, J. L. (1990). K-d Trees for Semidynamic Point Sets. *Sixth Annual Symposium on Computational Geometry* (S. 187-197). Berkley, California: ACM.

Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. (ACM, Hrsg.) *Communications of the ACM* , S. 509-517.

Bentley, J. L., & Friedman, J. H. (1979). Data Structures for Range Searching. *ACM Computing Surveys (CSUR)* (S. 397-409). New York, NY: ACM.

Buchanan, A., & Fitzgibbon, A. (2006). Interactive Feature Tracking using K-D Trees and Dynamic Programming. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (S. 626-633). IEEE Computer Society.

Coates, P., & Hazarika, L. (1999). The use of Genetic Programming for applications in the field of spatial composition. *Proceedings of the 2nd Generative Art Conference (GA1999).* Milan: Generative Design Lab Milan Polytechnic University, Italy.

Coates, P., Broughton, T., & Jackson, H. (1999). Exploring Three-dimensional design worlds using Lindenmeyer Systems and Genetic Programming. In P. Bentley, *Evolutionary Design Using Computers* (S. 323-341). San Francisco, California: Morgan Kaufmann Publishers Inc.

Coates, P., Derix, C., Krakhofer, S. P., & Karanouh, A. (2005). Generating architectural spatial configurations. Two approaches using Voronoi tessellations and particle systems. *Proceedings of the VIII Generative Art International Conference (GA2005).* Milan.

Coates, P., Healy, N., Lamb, C., & Voon, W. (1996). The use of Cellular Automata to explore bottom up architectonic rules. *Eurographics UK Chapter 14th Annual Conference.* London: Eurographics Association UK.

Coello Coello, C. A. (1998). A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems* , S. 269-308.

Cohon, J. L. (1978). *Multiobjective Programming and Planning.* New York, San Francisco, London: Academic Press.

Corne, D. W., Knowles, J. D., & Oates, M. J. (2000). The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature* (S. 839-848). London: Springer-Verlag.

De Berg, M., Cheong, O., Van Kreveld, M., & Overmars, M. (1997). *Computational Geometry: Algorithms and Applications.* Berlin: Springer-Verlag.

Duarte, J. P. (2001). Customizing Mass Housing: A Discursive Grammar for Siza's Malagueira Houses. *PhD Thesis* . Massachusetts Institute of Technology (MIT).

Elezkurtaj, T. (2004). Evolutionäre Algorithmen zur Unterstützung des kreativen architektonischen Entwerfens. *Dissertation* . Wien: Institut für Architekturwissenschaften, Technische Universität Wien.

Elezkurtaj, T., & Franck, G. (2002). Algorithmic Support of Creative Architectural Design. *Umbau, 19* , S. 129-137.

Frazer, J. (1995). *An Evolutionary Architecture.* London: Architectural Association.

Frazer, J. (April 1974). Reptiles. *Architectural Design* , S. 231-239.

Frew, R. S. (1980). A survey of space allocation algorithms in use in architectural design in the past twenty years. *DAC '80 Proceedings of the 17th Design Automation Conference* (S. 165-174). New York: ACM.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software (TOMS)* (S. 209-226). ACM.

Fussell, D., & Subramanian, K. R. (1988). Fast Ray Tracing Using K-d Trees. In *Technical Report: CS-TR-88-07.* Austin, TX: Department of Computer Sciences, University of Texas at Austin.

Goodrich, M. T., & Tamassia, R. (2002). *Algorithm Design: Foundations, Analysis and Internet Examples.* New York, NY: John Wiley & Sons.

Harding, J., & Derix, C. (2010). Associative Spatial Networks in Architectural Design: Artificial Cognition of Space using Neural Networks with Spectral Graph Theory. In J. S. Gero, *Design Computing and Cognition DCC'10* (S. 305-323). Stuttgart: Springer.

Horn, J. (1997). F1.9 Multicriterion decision making. In T. Bäck, D. B. Fogel, & Z. Michalewicz, *Handbook of Evolutionary Computation* (S. 1-9). Bristol: IOP Publishing Ltd.

Jo, J. H., & Gero, J. S. (1998). Space Layout Planning using an Evolutionary Approach. *Artificial Intelligence in Engineering, 12* , S. 149-162.

Kado, K. (1995). An Investigation of Genetic Algorithms for Facility Layout Problems. *Thesis* . Edinburgh: University of Edinburgh.

Kirsch, W. (1988). *Die Handhabung von Entscheidungsproblemen.* München: Verlag Barbara Kirsch.

Koenig, R., & Bauriedel, C. (2004). Computer-generated City Structures. *Generative Art Conference.* Milan.

Li, S.-P., Frazer, J., & Tang, M.-X. (2000). A Constraint-Based Generative System for Floor Layouts. *Fifth Conference on Computer Aided Architectural Design Research in Asia (CAADRIA).* Singapore.

Maneewongvatana, S., & Mount, D. M. (1999). It's okay to be skinny, if your friends are fat. *Center for Geometric Computing 4th Annual CGC Workshop on Computational Geometry.* John Hopkins University.

Moore, A. W. (1991). An introductory tutorial on kd-trees. *Extract from: Efficient Memory-based Learning for Robot Control Technical, PhD Thesis: Report No. 209* . Cambridge: University of Cambridge.

Rechenberg, I. (1994). *Evolutionsstrategie '94.* Stuttgart: Friedrich Frommann Verlag.

Schneider, S., Fischer, J.-R., & König, R. (2010). Rethinking Automated Layout Design: Developing a Creative Evolutionary Design Method for Layout Problems in Architecture and Urban Design. In J. Gero, *Design Computing and Cognition DCC'10* (S. 367-386). Stuttgart: Springer.

Wald, I., & Havran, V. (2006). On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N). *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, (S. 61-69). Salt Lake City, UT.

Whitley, D. (1994). A Genetic Algorithm Tutorial. *Statistics and Computing, Vol. 4, No.2* , S. 65-85.

Zitzler, E. (1999). Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. *TIK-SCHRIFTENREIHE NR. 30* . Zürich: Institut für Technische Informatik und Kommunikationsnetze, Eidgenössische Technische Hochschule Zürich.

Zitzler, E., Laumanns, M., & Bleuler, S. (2003). A Tutorial on Evolutionary Multiobjective Optimization. In X. Gandibleux, M. Sevaux, K. Sörensen, & V. T'Kindt, *In Metaheuristics for Multiobjective Optimisation* (S. 3-38). Springer-Verlag.